SERIE
AUTOAPRENDIZAJE
ACELERADO



ZX la mejor programación del Spectrum por la práctica

FE DE ERRATAS EN LAS PÁGINAS DE APÉNDICES

por Tim Hartnell y Dilwyn Jones

DESDE LAS PRIMERAS NOCIONES A LOS PROGRAMAS MAS COMPLEJOS, A TRAVES DE UNA FORMACION PROGRESIVA

Más de 100 programas y rutinas que FUNCIONAN





TIM HARTNELL - DILWYN JONES

LA MEJOR PROGRAMACION DEL ZX SPECTRUM POR LA PRACTICA

EDICIONES TECNICAS REDE, S.A.
Apartado 35.400
BARCELONA

Publicado en Gran Bretaña por
INTERFACE, London
con el título de «PROGRAMMING YOUR ZX SPECTRUM»

Traducción: Alfonso Martínez María

Todos los programas han sido comprobados en el Departamento de Microinformática de REDE.

Este libro está dedicado a Iris

ISBN: 84-247-0195-X

Impreso en España | Printed in Spain

Dep. Legal: B. 16783/1984 | —REDEPRINT— | Barcelona

Sumario

Prólogo

Las primeras horas con su ZX Spectrum pueden producirle aturdimiento. Después de haber ejecutado los programas del manual ofrecidos como muestra, es posible que piense: "Si, ¿pero ahora qué?". Este libro pretende responder a la pregunta. Le llevará de la mano, programando el Spectrum desde principios elementales hasta complicadas técnicas de programación.

Y mientras vaya ejecutando programas, destruyendo alienígenos y asteroides por todas partes, descubrirá que está aprendiendo a programar a la vez que adquiere conocimientos sobre los ordenadores en general sin ningún esfuerzo.

Este libro intenta ser una herramienta para tenerla a su lado cuando se siente frente al ordenador. Su valor quedará notablemente disminuido si solamente trata de leer los programas. Sería mejor que ejecutase cada uno de ellos a medida que va llegando a ellos sin "dejar vivo a ningún alienígeno". De esta forma logrará el máximo rendimiento del libro y se habrá convertido en un "genio" de la programación antes de saber dónde se encuentra.

Es el momento de empezar. Conecte su Spectrum y el televisor, y comencemos.

TIM HARTNELL, Londres
DILWYN JONES, Bangor, Gwynedd

La utilización del teclado

Aunque, a primera vista, el teclado del Spectrum pueda parecerle complicado, no es muy difícil dominarlo siempre que proceda cuidadosamente en los primeros pasos.

Las dos teclas más importantes son las de cambio, "CAPS SHIFT" (esquina inferior izquierda) y "SYMBOL SHIFT" (la segunda de la derecha de la última hilera). Búsquelas ahora. Los colores de estas teclas indican una de sus aplicaciones. Conecte su Spectrum, mantenga presionada la "CAPS SHIFT" y pulse cualquiera de las teclas con letras del alfabeto. Verá que aparece la mayúscula. "CAPS SHIFT" también reproduce las palabras en blanco encima de las teclas del 1 al 4. Mantenga presionada "CAPS SHIFT" y pulse la tecla 2. Ahora accione cualquiera de las letras y verá cómo se presentan en la versión mayúscula. El uso de "TRUE VIDEO" e "INVERSE VIDEO" se discutirá en la parte relativa a gráficos.

Dejando la tecla blanca "SHIFT", veamos ahora la roja "SYMBOL SHIFT". Manténgala apretada y presione cualquier tecla (excepto ENTER, BREAK SPACE o CAPS SHIFT). Observará que se presentan los pequeños símbolos rojos de la tecla (como la flecha de la tecla H, apuntando hacia arriba).

A continuación pulse cualquier tecla alfabética sin mantener presionada ninguna de las "SHIFTS". Verá que sale la palabra blanca (IF, NEXT, DIM, etc.). Estas son *claves* ("Keywords") que constituyen las primeras palabras de una línea de programa y de las que trataremos próximamente.

Se logran las palabras en verde, que hay encima de las teclas, presionando ambas "SHIFT" simultáneamente, dejando la pulsación y accionando la tecla cuya leyenda se desea. Por ejemplo, mantenga apretadas ambas "SHIFTS", suéltelas y pulse la D. Aparecerá la palabra "DATA" (lo que se indica encima de la tecla).

Las palabras rojas debajo de cada tecla se obtienen presionando simultáneamente ambos "SHIFTS", soltando el blanco y conservando la presión sobre el rojo mientras se pulsa la tecla deseada. Se logrará la palabra "BRIGHT" pulsando así la tecla B, "ATTR" con la L y "VERIFY" con la R. Lo dicho vale para todo excepto para EDIT, ENTER, GRAPHICS y DELETE.

EDIT. Se utiliza para modificar una línea del programa. Poniendo el cursor en tal línea (el signo "mayor que" >) y presionando la tecla "1/EDIT", mientras se mantiene apretada la "CAPS SHIFT" se llevará la línea que se va a "editar" al fondo de la pantalla. Las teclas 5 y 8 le llevarán a lo largo de la línea en la dirección de la punta de flecha que se muestra en estas teclas.

ENTER. Se presiona esta tecla después de escribir una línea del programa en el fondo de la pantalla para hacer que entre en la parte principal del programa, en la parte superior de aquélla. También se utiliza después de una palabra como "RUN" para hacer que el ordenador ejecute la instrucción.

GRAPHICS. Si se mantiene apretada "CAPS SHIFT" y se pulsa la tecla 9, se verá cómo la letra "cursor" de la pantalla se convierte en una G. En ese momento, accione cualquiera de las teclas de números y vea lo que aparece en la pantalla. Mantenga apretada la "CAPS SHIFT" y pulse cualquiera de aquéllas: verá cómo se reproduce el gráfico "opuesto" del que teníamos sin pulsar "CAPS SHIFT". La segunda utilidad de la modalidad "GRAPHICS", para las teclas a definir por el usuario, se tratará más adelante.

DELETE. Esta es una tecla de "borrar". Apriete y mantenga la "CAPS SHIFT", presione la tecla 0 y se borrará, elemento por elemento, la línea en que está trabajando. Puede levantar el dedo de la tecla "CAPS SHIFT" cuando la acción de borrar es repetitiva y con sólo "DELETE" presionada continuará la acción de borrado.

No se preocupe si esta descripción, que se ha hecho lo más sencilla y clara que hemos podido, no le descubre inmediatamente todos los misterios del teclado. Utilizando su Spectrum y hallando cada cosa a medida que se necesite, llegará indudablemente al punto en que lo utilice perfectamente.

La instrucción PRINT

PRINT es probablemente la instrucción más utilizada en el lenguaje BASIC. Permite al ordenador comunicarse con usted. Escriba la línea siguiente en su Spectrum y después pulse la tecla ENTER:

PRINT 5

Observará que el ordenador obedientemente presenta el número 5. Puede utilizar la instrucción PRINT para que el ordenador calcule. Escriba lo siguiente y a continuación pulse ENTER:

PRINT 5-3

Al pulsar ENTER observará el resultado correcto. Esta modalidad de cálculo directo puede resolver problemas tan complejos como desee. Pruebe lo siguiente, recordando presionar ENTER después para que el ordenador actúe sobre lo expresado:

PRINT SQR (8 + 1)

Con ello se pide al ordenador que calcule la raíz cuadrada (esto es lo que significa SQR) de la suma de los números encerrados entre paréntesis, es decir, la raíz cuadrada de nueve. Si el ordenador funciona correctamente conseguirá, por supuesto, el valor tres.

Por consiguiente, puede ver que PRINT actúa en forma directa para presentar números y resultados del cálculo. También puede presentar palabras. Accione la tecla de fijación de mayúsculas (CAPS LOCK) mientras se mantiene pulsada la CAPS SHIFT y, al propio tiempo, pulse la tecla 2. Las letras aparecerán todas mayúsculas. Pruebe lo siguiente y después pulse ENTER:

PRINT HOLA AMIGOS

En lugar de ofrecernos alegremente el mensaje "HOLA AMIGOS" el ordenador nos viene con lo que se llama "mensaje de error". En este caso, este mensaje dice "2 variable not found" ("no encuentra la variable"). Si desea que se escriban palabras han de ponerse entre comillas. Instruya y ejecute (es decir, pulse ENTER después de escribir) lo siguiente:

PRINT "HOLA AMIGOS"

Observará que el mensaje aparece escrito en la parte superior de la pantalla.

Recapitulemos rápidamente. Utilizando simplemente como una instrucción, PRINT 2 + 3 le dirá al ordenador que presente el resultado de la suma. Si la instrucción es PRINT "PALABRAS" haremos que en la pantalla aparezca todo lo escrito entre las comillas.

Los ordenadores utilizan programas y ya ha llegado el momento

de escribir nuestro primer programa sencillo. Entre y ejecute el que sigue. Cuando lo haga, para lo que pulsará el enunciado RUN (tecla R) y ENTER, verá algo similar a lo que aparece debajo del listado del programa.

```
   10 REM PROGRAMA UNO
   20 PRINT "ESTO ES UNA DEMOSTRA
CION"
   30 PRINT 1
   40 PRINT 2
   50 PRINT "FIN"

ESTO ES UNA DEMOSTRACION
1
2
FIN
```

Mientras tenemos este programa en el ordenador, aprendamos algo más sobre programación. Entremos la palabra LIST (que se hace apretando la tecla K) y después presionemos ENTER. Observará que el listado del programa vuelve a aparecer. Advierta que cada línea empieza con un número. La primera, numerada 10 en este caso, se inicia con la palabra REM, que en el lenguaje del ordenador significa "Nota" y se utiliza en un programa cuando se desea explicar lo que hay en él o lo que es (como en este caso), de forma que, cuando posteriormente se vuelva a este programa, se sepa de qué trata. El ordenador ignora las instrucciones REM.

Una instrucción REM se compone de un número de línea, después la palabra REM y, por último, algún texto. El mensaje que sigue a la palabra REM puede componerse de cualquier forma que le parezca —letras, números, signos de puntuación, gráficos o espacios— aunque es mejor que sea lo más breve y claro posible. Si bien lo que se escribe después de REM es ignorado por el ordenador al ejecutar el programa, no por eso deja de ocupar un espacio de memoria.

Las instrucciones REM son frecuentemente del tipo siguiente:

```
10 REM ESTO CALCULA EL TANTEO
10 REM ENCUENTRA EL ANGULO
```

No hay ninguna razón para que sólo exista una instrucción REM pero, si el comentario que se desea añadir a una línea de programa precisa más de una línea de texto, es importante poner la clave REM al principio de cada una. Por ejemplo:

```
  60 REM LA RUTINA DE MULTIPLICA
CION POR LA QUE
  70 REM LAS DOS VARIABLES A Y B
  80 REM SE MULTIPLICAN CONJUNTA
MENTE
```

En cualquier línea que empiece con la clave REM, el texto que sigue a la misma será ignorado por el ordenador. No obstante, el listado completo del programa, incluyendo los mensajes REM, aparecerán en la pantalla si se solicita su presentación mediante LIST.

Echemos ahora una ojeada a la edición. Escriba el número 10 y pulse ENTER. La línea de este número ha desaparecido. Es muy fácil desembarazarse de las que no se desean. Basta con escribir el número que la encabeza y dar la orden de ejecución con el ENTER.

```
  20 PRINT "ESTO ES UNA DEMOSTRA
CION"
  30 PRINT 1
  40 PRINT 2
  50 PRINT "FIN"
```

Añada 10 REM y pulse ENTER.

Recordará, de las veces que ha pulsado LIST en los trabajos de esta sección, que LIST es la instrucción BASIC que utilizamos para hacer que el ordenador presente la totalidad del listado del programa que está trabajando. Todas sus líneas se relacionan por orden numérico, en lugar de hacerlo por el que fueron introducidas en el ordenador. Es decir, que la máquina pone automáticamente las líneas en orden. Escriba lo que sigue y pulse ENTER:

15 PRINT "LINEA NUEVA" (NEWLINE)

Observará en la nueva presentación del programa que esta línea (15) se pone automáticamente en su posición correcta en el listado.

```
  10 REM
  15 PRINT "ESTO ES UNA LINEA NU
EVA"
  20 PRINT "ESTO ES UNA DEMOSTRA
CION"
  30 PRINT 1
  40 PRINT 2
  50 PRINT "FIN"
```

Como sin duda habrá comprendido, la instrucción o enunciado RUN se utiliza para que el ordenador empiece a ejecutar un programa que le ha sido aplicado, bien escribiéndolo con el teclado o cargán-

dolo con una cinta magnetofónica grabada ("cassette"). El ordenador ejecuta todas las líneas que tiene almacenadas en su memoria, empezando por la de número más bajo, y continuando por el orden numérico. Varias instrucciones hacen que la ejecución retroceda hasta un punto anterior del programa y vuelva a repetirse (bucles) pero, en esencia, el ordenador sigue el programa según el orden del número de la línea, a menos que se le diga otra cosa.

Si se desea que el programa se detenga en un punto determinado, puede utilizarse una instrucción denominada STOP. Teclee 25 STOP (con A, manteniendo pulsado el SHIFT rojo) y aplique el ENTER. A continuación ejecute el programa. En la pantalla se verá:

LINEA NUEVA
ESTO ES UNA DEMOSTRACIÓN

Y en el fondo de la pantalla se leerá el mensaje "9 STOP statement 25:1" que significa la ejecución del STOP por la primera instrucción de la línea 25.

Volveremos al comando PRINT con un poco más de detalle más adelante pero existe otra instrucción que me gustaría introducir en este momento. El enunciado NEW borra cualquier programa de la memoria del ordenador y debe utilizarse para eliminar de ella toda información antes de empezar a introducir un nuevo programa. Si no se hiciese esto y se empleasen números de línea diferentes se entremezclarían las nuevas con las anteriores del viejo programa. La instrucción NEW es brutal y definitiva haciendo que el ordenador olvide todo lo que había recibido directamente o a través de cinta magnética.

Pruébelo en su ordenador. Escriba NEW (con la A) y pulse ENTER; a continuación aplique LIST y nuevamente ENTER. Encontrará naturalmente que no aparece ningún listado. Pruebe LIST 10 y conseguirá el mismo resultado negativo.

Las instrucciones PRINT y TAB

Para continuar nuestra exploración de la instrucción PRINT, bloquee las mayúsculas ("CAPS LOCK") como antes y aplique y ejecute el siguiente programa.

```
  10 REM PRESENTACION DE FORMATO
S
  20 PRINT
  30 PRINT
  40 PRINT
  50 PRINT
  60 PRINT "HOLA ";60
  70 PRINT ,"HOLA ";70
  80 PRINT 12
  90 PRINT 1,2
 100 PRINT ,1
 110 PRINT ,1;2
```

Siga esta explicación cuidadosamente y aprenderá mucho sobre cómo el ordenador compone su presentación de salida. Podrá aplicar lo que ha aprendido para organizar sus propios programas según su deseo. Seguiremos el programa línea por línea:

10 Título, instrucción REM.

20-50 Cada una de estas instrucciones PRINT, sin nada a continuación, deja una línea que no contiene ninguna inscripción. Esto explica el espacio que queda en la parte superior de la pantalla, cuando ejecute el programa, antes de que algo aparezca escrito.

60 Esta línea hace presentar la palabra HOLA y, dejando un espacio, aparece el número 60 para saber de qué línea procede.

70 La coma (el SHIFT rojo y la tecla N, cerca de la esquina inferior derecha del teclado), como habrá apreciado, desplaza el origen de la línea hasta el centro de la pantalla.

80 Esta instrucción permite que los números 1 y 2 se presenten juntos. Adviértase que aunque existiera un espacio entre ambos al redactar el programa (como en PRINT 1 2), el ordenador los seguiría poniendo juntos, es decir, como 12.

90 Esta línea utiliza comas entre los números para asegurarse que se presentan separadamente en ambas mitades de la pantalla.

100 La coma al principio de la línea desplaza el 1 hasta la segunda mitad, como en el caso de la palabra HOLA de la instrucción 70.

110 El punto y coma entre los números asegura que se presenten juntos como en la línea 80.

Pueden utilizarse la coma y el punto y coma en las instrucciones PRINT para controlar la presentación en la pantalla que se desea. Borre

el programa con NEW y después aplique y ejecute la siguiente serie de programas para producir cierto número de efectos.

El primer programa, denominado PRINT DOS, simplemente presenta los números del 1 al 10, en el lado inferior de la pantalla. El siguiente (PRINT DOS -B), los presenta apretadamente juntos en una línea. El PRINT DOS-C, en pequeñas columnas; y el PRINT DOS-D imprime nuevamente los números del 1 al 10 con un solo espacio entre ellos.

```
10 REM ESCRIBE DOS
20 FOR J=1 TO 10
30 PRINT J
40 NEXT J
```

```
10 REM ESCRIBE DOS-B
20 FOR J=1 TO 10
30 PRINT J;
40 NEXT J
```

```
10 REM ESCRIBE DOS-C
20 FOR J=1 TO 10
30 PRINT J,
40 NEXT J
```

```
10 REM ESCRIBE DOS-D
20 FOR J=1 TO 10
30 PRINT J;" ";
40 NEXT J
```

La utilización de TAB

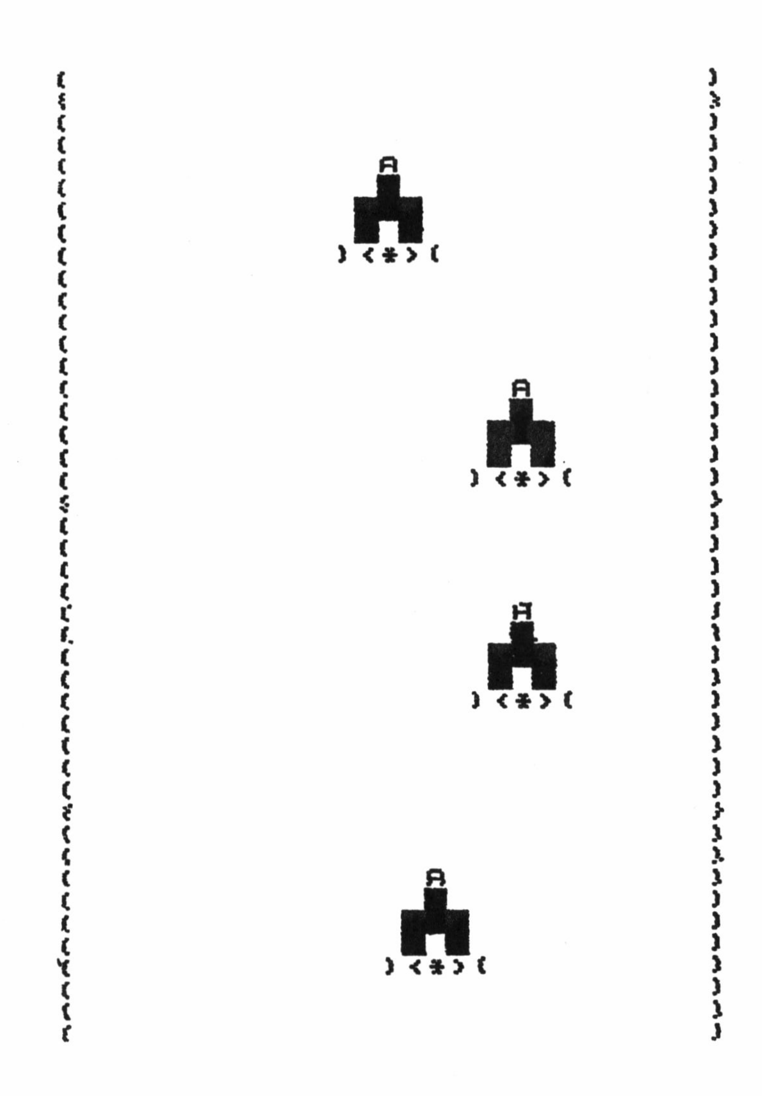
El enunciado TAB (de tabulación) puede combinarse ventajosamente con el PRINT. Desplaza la posición PRINT de la presentación el número de espacios especificados con un número. Aplique los programas PRINT DOS-E y PRINT DOS-F y vea los efectos de la instrucción TAB en ellos.

```
10 REM ESCRIBE DOS-E
20 FOR J=1 TO 10
30 PRINT TAB J;J
40 NEXT J
```

```
10 REM ESCRIBE DOS-F
20 FOR J=1 TO 10
30 PRINT TAB 3*J;J
40 NEXT J
```

El siguiente programa TABULADOR LANZAMIENTO DE COHETES muestra la eficacia con la que puede utilizarse la instrucción TAB. Aplíquelo y ejecútelo. Después, fije de nuevo la atención en el libro para analizar sus líneas más importantes.

```
   1 PAPER 7: CLS
  10 REM TABULADOR LANZAMIENTO
DE COHETES
  20 DIM A$(6,6)
  25 BORDER 0
  30 FOR J=10 TO 1 STEP -1
  40 PRINT TAB 3*J; INK 2;J
  50 FOR A=1 TO J
  52 BEEP .02,5*J
  60 NEXT A
  70 NEXT J
  71 FOR B=1 TO 6
  72 READ Q$
  73 LET A$(B)=Q$
  74 NEXT B
  80 REM ** PROGRAMA PRINCIPAL *
*
  90 LET Q=INT (RND*23)+1
 100 FOR R=1 TO 6
 110 BEEP .02,10*R
 120 PRINT "(";TAB Q; INK R;A$(R
); INK 0;TAB 30;")"
 130 NEXT R
 140 POKE 23692,-1
 150 LET ESPACIO=Q/3
 180 FOR P=1 TO ESPACIO
 190 PRINT "(";TAB 30;")"
 200 NEXT P
 210 GO TO 90
 220 DATA "  A","  █"," ███"," █
█",")<*>(",""
```



Las líneas más útiles para este estudio son la 120 y 190 ya que emplean la instrucción TAB para la presentación. La línea 120 se comporta como sigue:

"("	Imprime el paréntesis izquierdo junto al borde de este lado de la pantalla.
TAB 9	El 9 es un número entre uno y 25 (seleccionado en la línea 90) que determina cuántos espacios laterales se moverá la posición PRINT.

INK r	Determina el color con que se va a representar cada par de cohetes.
A$(r)	Indica la parte del cohete que se va a representar. Utiliza elementos de la variable en cadena A$ que se asignó por el bucle (loop) READ de las instrucciones 71 a 74. No se preocupe por esto ahora; se tratará en detalle más adelante.
INK 0	Devuelve el color de INK a negro.
TAB (30)	Después de que se ha representado la parte del cohete de la línea, la posición PRINT se desplaza a la posición 31 de la línea, donde se imprime ")" para situar un límite inferior en el lado derecho de la pantalla del límite ")".

Veamos ahora la línea 190:

La línea se encuentra dentro del bucle que empieza en la 180 y termina en la 200. El enunciado PRINT "("; TAB 30;")"imprime")" en el lado izquierdo de la pantalla y después se traslada a la posición 31 (mediante TAB 30) para colocar un ")" en el lado derecho. La línea 190 utiliza un número aleatorio de veces determinado por 9 que ha sido seleccionado en la línea 90) para establecer un número aleatorio de líneas vacías entre los sucesivos cohetes para espaciarlos. La línea 140(POKE 23692,-1) asegura que el programa no va deteniéndose periódicamente para preguntar si continúa ("scroll?").

Conservación de programas (SAVE)

Puede ser interesante conservar una copia permanente del programa TABULADOR LANZAMIENTO DE COHETES. Los programas podrá conservarlos introduciéndolos en el ordenador mediante el accionamiento del teclado y previa conexión del cassette, de acuerdo con las instrucciones de su manual. Después pulse SAVE seguido del nombre del programa (puesto entre comillas). En este caso, le sugerimos que le llame COHETES, de forma que la instrucción sea

SAVE "COHETES". Ponga en marcha el cassette, con la tecla de grabación apretada (siguiendo las instrucciones del manual, repetimos) y después pulse ENTER.

Sugerimos que se adquiera el hábito de conservar cada programa tres veces seguidas en la misma cinta (C-12 ó C-15 para ordenadores) y que sólo se grabe un programa en cada cara de la cinta. Póngase una etiqueta con el nombre del programa (COHETES, en este caso). Aunque pudiera parecer un derroche utilizar una cara entera de la cinta con un solo programa (grabado tres veces), le ahorrará tiempo al no tener que buscarlo, lo que constituye un proceso engorroso, y le compensará sobradamente del gasto de utilizar más cassettes que las estrictamente necesarias. Se graba tres veces el programa para el caso de que la cinta se dañe en algún punto o, como a veces sucede, una grabación no se carga adecuadamente.

Deben limpiarse frecuentemente las cabezas de grabación mediante un producto líquido o con una cassette de cinta limpiadora. Se trata de asegurarse que la grabación produzca la señal más clara posible.

La comprobación de la buena grabación de un programa

Una vez se tenga un programa grabado en cinta, se puede comprobar que ha entrado correctamente antes de borrarlo de la memoria del ordenador.

Ya hemos visto que para grabar se ponía SAVE "PROGRAMA", se accionaba la grabadora y se pulsaba cualquier letra del teclado del ordenador. Para comprobar la grabación, se rebobina la cinta, se pone el enunciado VERIFY "PROGRAMA", se pulsa ENTER y se inicia la reproducción del contenido. Si todo está bien, aparecerá en la pantalla el nombre del programa y, al final, el mensaje 0 OK. A partir de este momento sabremos que el programa está completamente grabado en la cinta.

La combinación de dos programas en el ordenador

Es posible cargar un nuevo programa en el ordenador mientras se conserva todavía el antiguo. Ambos programas se combinarán. Si los dos tienen idénticos números de línea, las instrucciones del nuevo harán desaparecer las del otro. Veamos un ejemplo. Introducimos el programa que sigue, lo grabamos (SAVE) y lo eliminamos de la memoria del ordenador (NEW).

```
10 REM PROGRAMA DE PRUEBA UNO
20 REM LINEA 20
30 REM LINEA 30
40 REM LINEA 40
```

Después se aplica al ordenador este otro programa que denominamos FUSION "UNO". El primero de los programas se había conservado con el nombre de "UNO" y, a continuación, se carga, mediante el magnetófono aquel primer programa grabado. El segundo programa es:

```
 5 REM PROGRAMA DOS
15 REM LINEA 15
25 REM LINEA 25
35 REM LINEA 35
```

En unos segundos nos aparece el siguiente programa en pantalla:

```
 5 REM PROGRAMA DOS
10 REM PROGRAMA DE PRUEBA UNO
15 REM LINEA 15
20 REM LINEA 20
25 REM LINEA 25
30 REM LINEA 30
35 REM LINEA 35
40 REM LINEA 40
```

El enunciado MERGE (fusionar) es muy útil para aprovechar las rutinas especiales, como la de RENUMERAR que se puede cargar después de haber introducido otro programa. Vale la pena asegu-

rarse que las rutinas que se empleen tengan los números de línea elevados (digamos que por encima de 9000) y que los inferiores sean utilizados para el programa. Esto asegura que no haya problemas de líneas duplicadas.

Consejos para resolver problemas en la carga de programas

En ocasiones puede resultar difícil el proceso de volver a llevar los programas grabados al ordenador. Si existen problemas de carga (LOAD) pruébense las siguientes sugerencias:

(I) Desconéctese el cable que no se usa de los dos que unen el ordenador y el magnetófono.
(II) Inténtese hacer funcionar el magnetófono con la alimentación de pilas.
(III) Sepárense el magnetófono, el ordenador y el televisor tanto como sea posible.
(IV) Cámbiese el nivel de volumen del magnetófono ya que de unos a otros puede variar la potencia de la salida. Pruébese el cambio de los niveles del control de tono, elevando, en particular, el de los agudos y reduciendo el de los bajos.
(V) Asegúrese de que los cables no están rotos y de que en los mismos no hay soldaduras desprendidas.
(VI) Esto parece tonto, pero hay que asegurarse de que las clavijas están correctamente colocadas en sus respectivos alojamientos. Puede ser útil poner unas etiquetas en el ordenador por encima de los correspondientes zócalos que nos permitan conocer su situación sin tener que mirar cada vez en la parte de atrás.

Regresemos ahora a TAB.

Sólo podemos utilizar TAB con un solo número después de la palabra. Recuérdese, TAB A desplazará A + 1 espacios laterales el principio de la posición PRINT en la línea. También podemos tener la palabra PRINT seguida por AT, y dos números, como PRINT AT 10,6;

que moverá la posición PRINT siete espacios laterales y once hacia abajo. La esquina superior izquierda de la pantalla tiene asignado el valor cero,cero, por lo que la instrucción PRINT AT 0,0; indica que la representación empezará en la referida esquina de la pantalla. Todo su lado izquierdo corresponde a 0, mientras que el derecho es el 31. La pantalla tiene un ancho de 32 caracteres, así que la posición más alejada de la derecha se numera con el 31.

PRINT AT

El siguiente programa "SQUASH" utiliza PRINT AT Y,X para poner en posición una bola (línea 620) y un bate (línea 150). Se emplean las teclas "Z" y "M" para mover el bate en el fondo de la pantalla, a derecha e izquierda respectivamente. El programa cuenta el tiempo que se mantiene la bola en juego y proporciona una puntuación en función de aquél. Pulsando cualquier tecla, al final del juego, se puede jugar de nuevo.

El listado puede parecer horripilante de momento pero una vez se haya familiarizado con estas tareas a lo largo de este libro, puede desear volver a programas como éste para desentrañar qué hace cada una de sus partes. Quedará sorprendido de lo mucho que podrá descifrar.

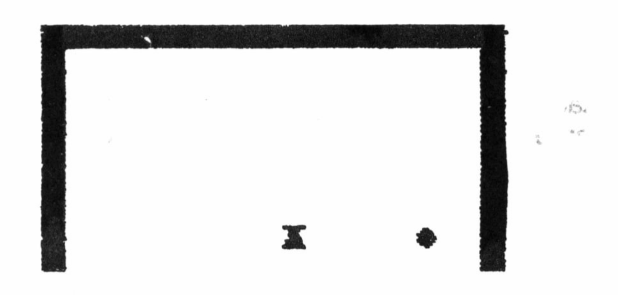
```
  10 REM SQUASH
  15 REM PROGRAMA DE
            JEREMY RUSTON
  20 GO SUB 690
  25 REM DESPLACE EL BATE CON
LAS TECLAS Z Y M
  27 LET DESPLAZAMIENTO=550
  30 LET SETUP=300
  35 LET MOV BATE=460
  40 BRIGHT 1: BORDER 4
  45 PAPER 7: CLS
  50 LET PUNTUACION=0
  60 GO SUB SETUP
  70 REM **************
  80 LET PUNTUACION=PUNTUACION+
INCREMENTO
 110 LET A$=INKEY$
 130 IF A$="Z" OR A$="M" THEN GO
 SUB MOV BATE
```

```
 140 GO SUB DESPLAZAMIENTO
 150 PRINT AT 19,B+11; INK 2;B$
 160 GO TO 80
 290 REM ****************
 300 REM *** SET UP ****
 310 LET X=1
 320 PRINT AT 10,10; INK 1;"████
█████████████████"
 330 FOR T=0 TO 10
 340 PRINT AT T+10,10; INK 1;"█"
;AT T+10,30;"█"
 350 NEXT T
 360 LET B$=" "+B$+" "
 380 LET Y=1
 385 LET L=1
 390 LET M=1
 400 LET B=10
 410 PRINT AT 19,11+B;B$
 420 LET INCREMENTO=207+INT (RND
*100)
 430 RETURN
 450 REM ****************
 460 REM ** MOV BATE ****
 480 IF A$="M" AND B=16 THEN
RETURN
 490 IF A$="Z" AND B=0  THEN
RETURN
 510 IF A$="M" THEN LET B=B+1
 520 IF A$="Z" THEN LET B=B-1
 530 RETURN
 540 REM ********************
 550 REM ** DESPLAZAMIENTO **
 570 PRINT AT 11+Y,11+X;" "
 580 IF L+X>18 OR L+X<0 THEN LET
 L=-L: BEEP .01,50
 590 IF M+Y>8 OR M+Y<0 THEN LET
M=-M: BEEP .01,20
 600 LET X=X+L
 610 LET Y=Y+M
 620 PRINT AT 11+Y,11+X; INK 4;D
$
 622 IF Y<>8 THEN RETURN
 625 PRINT AT 6,7; INK 6; PAPER
2;" EL TANTEO ES ";PUNTUACION;"
"
 630 IF Y=8 AND ABS (B-X)<=2
THEN RETURN
 640 PRINT AT 2,2; INK 7; PAPER
3;" FIN DEL JUEGO "
 650 FOR G=1 TO 50
 660 BEEP .01,G: BEEP .01,60-G
 670 NEXT G
 680 RUN
 690 REM LA LETRA ENTRE COMILLAS
 EN LA LINEA 700 ES UNA M,PUESTA
 DESPUES DE ADOPTAR LA MODALIDAD
 DE GRAFICOS.
 700 LET D$="●"
 710 FOR H=0 TO 7
```

```
 715 BEEP .01,H*H
 720 READ N
 730 POKE USR "●"+H,N
 740 NEXT H
 750 REM LA SIGUIENTE LETRA ES
LA REPRESENTACION GRAFICA DE B.
 760 LET B$="⧗"
 770 FOR H=0 TO 7
 775 BEEP .01,50/(H+1)
 780 READ N
 790 POKE USR "⧗"+H,N
 800 NEXT H
 890 RETURN
1000 DATA BIN 00000000,BIN 00011
000,BIN 00111100,BIN 001111110,B
IN 001111110,BIN 001111110,BIN 0
0111100,BIN 00001000
1010 DATA BIN 11111111,BIN 00111
100,BIN 00011000,BIN 00111100,BI
N 0111100,BIN 01111110,BIN 01111
110,BIN 11111111
```

```
FIN DEL JUEGO


      EL TANTEO ES 16665
```



Los colores y los gráficos

El Spectrum está equipado con poderosas instrucciones gráficas que pueden utilizarse para dar más realce a sus programas. Tales instrucciones son de uso sencillo y capaces de producir una amplia escala de efectos.

Hay tres cosas que se pueden controlar con el mando del color: el cerco o límite alrededor de la zona principal de presentación (que se consigue con la instrucción BORDER), la mencionada zona de presentación (controlada con PAPER) y los colores de la presentación (INK).

Existen ocho colores disponibles (si se cuentan el blanco y el negro como tales) y están numerados de cero a siete. Son los siguientes:

0 — negro
1 — azul
2 — rojo
3 — magenta (púrpura)
4 — verde
5 — cyan
6 — amarillo
7 — blanco

Cuanto más bajo es el número, más oscuro es el color. En un televisor en blanco y negro, los números inferiores están próximos al negro y los superiores al blanco.

Cuando conecte su Spectrum, por primera vez, tendrá el centro de la pantalla (PAPER) blanco así como el margen (BORDER), y las presentaciones (INK) serán negras. Es decir, la pantalla estará completamente blanca y cualquier programa que se introduzca aparecerá en negro.

Los colores de tinta y fondo (INK y PAPER) se pueden utilizar de dos maneras. La primera es en forma global; es decir, si una línea de programa dice PAPER 6, seguido de CLS (despejar la pantalla), el fondo (la zona encuadrada por el margen) se volverá amarillo. De forma similar, una línea de programa INK 2 asegura que la representación en aquel punto será roja.

También es posible utilizar los colores en zonas limitadas. Si la instrucción es PRINT INK1;PAPER 7; "HOLA AMIGOS", el Spectrum presentará las palabras entre comillas en blanco sobre una pequeña banda azul. El mismo control de una zona concreta es posible en las INPUT. Si se desea una cadena como dato de entrada, se puede hacer, por ejemplo, INPUT (INK 2; PAPER 6; "Cuál es su nombre"); la frase se presentaría impresa en rojo sobre una pequeña banda amarilla.

Comprobemos ahora las instrucciones de color mediante el siguiente programa:

```
20 REM DEMOSTRACION DEL COLOR
30 FOR B=0 TO 7
40 FOR P=0 TO 7
50 FOR I=0 TO 7
60 BORDER B
70 PAPER P: CLS
80 INK I
90 PRINT AT 10,10;"MARGEN(BORD
ER) ";B;TAB 10;"FONDO( PAPER ) "
;P;TAB 10;"TINTA( INK ) ";I
100 FOR U=1 TO 60: NEXT U: BEEP
.15,P*I+2*B
110 NEXT I
120 NEXT P
130 NEXT B
```

Este programa representa todas las combinaciones de MARGEN, PAPEL y TINTA (BORDER, PAPER, INK). Como verá necesita un largo tiempo para su ejecución (existen 8 * 8 * 8 combinaciones posibles) aunque varias de ellas no son muy efectivas (tinta blanca sobre papel y margen del mismo color no resulta precisamente muy fácil de leer). Otras combinaciones no son atractivas si bien hay algunas que son verdaderamente muy interesantes y vale la pena que tenga a mano papel y lápiz mientras ejecuta el programa para tomar nota de cuáles son.

La línea que despeja la pantalla (70 CLS) es necesaria para hacer que el color del papel se aplique en forma "global". Con ello sólo cambia el fondo debajo de las palabras que se imprimen. Pase el programa otra vez sin la línea 40. Las instrucciones de tintas (INK) utilizadas en un listado, que tienen automáticamente efectos globales si se hallan en una línea separada seguidas de CLS, limitan su acción a una zona determinada si son seguidas de PRINT o INPUT. Una instrucción global INK (como INK 2 para que toda la impresión sea roja) no cambia por otra limitada (como PRINT INK 1;"test") ya que el color revierte al globalmente definido en cuanto aparece en el programa una instrucción PRINT sin parámetro INK.

Ejecute ahora el siguiente programa que muestra la eficacia con que se mezclan los colores cuando se eligen aleatoriamente.

```
6 REM PIRAMIDE
7 REM © HUGHES, HARTNELL
10 BORDER 7
15 CLS
20 LET B=16
50 LET T=0
60 LET S=0
70 LET L=20
```

```
 80 LET T=T+3
 90 FOR N=S TO S+B*2-2
100 PRINT AT L,N; INK INT (RND*
6)+1;"█"
105 BORDER INT (RND*6)+1
110 NEXT N
120 LET L=L-1
130 LET B=B-1
140 LET S=S+1
150 IF B>0 THEN GO TO 80
155 BORDER 1
160 GO TO 160
```

El programa dibuja una pirámide de pequeños bloques coloreados. El margen (BORDER) produce destellos en forma alarmante durante todo el tiempo, y finalmente (línea 155) se hace azul. La línea 160, que se llama a sí misma, tiene por objeto suprimir el "OK" que, de otra forma, estropearía la presentación. Se sale del programa pulsando "BREAK".

El pequeño cuadrado negro al final de la línea 100 se obtiene directamente del teclado en modo gráficos ("GRAPHICS") (se presiona tecla SHIFT blanca y se pulsa la 9), pulsando después la tecla 8 mientras se mantiene presionada la SHIFT BLANCA. Los inversos de otros caracteres se logran pulsando "INV.VIDEO" (SHIFT blanco y la 4). Se vuelve a lo que se llama "TRUE VIDEO" (video verdadero) apretando la tecla SHIFT blanca y la 3. El fondo negro de detrás de las letras inversas se hace del color de la tinta (INK) y las propias letras toman el color del fondo de la pantalla, lo que resulta más efectivo como demuestra el siguiente programa:

```
15 PAPER 5
17 CLS
20 FOR G=1 TO 100
30 INK RND*7
40 PAPER RND*7
50 BORDER RND*7
70 PRINT AT RND*21,RND*9;"ESTO
 ES UNA DEMOSTRACION"
80 NEXT G
```

Utilizando este listado de colores directamente en un programa puede producir buenos resultados como el siguiente CODIGO DE COLORES demuestra. Es una variación del "Mastermind", como verá si lo ejecuta. El programa espera que adivine un código de cuatro colores, no cuatro números o letras como en la mayoría de otras

versiones del juego. Aplique y ejecute el programa. Después vuelva al libro para la explicación de las instrucciones que se utilizan sobre colores y gráficos.

```
  10 REM CODIGO DE COLORES
  20 POKE 23609,100
  30 DIM C(4)
  40 DIM G(4)
  50 DIM H(4)
  60 INK 0: BORDER 0: PAPER 7:
CLS
  70 PRINT ''TAB 3;"ESTOY PENSAN
DO EN UN CODIGO"
  80 PRINT "DE 4 COLORES. TIENES
 10 INTENTOS"
  90 PRINT "PARA ACERTARLO. ESCO
JO ENTRE    ESTOS COLORES"
 100 FOR C=1 TO 6
 110 PRINT TAB 4+C; INK 0;C;;" >
>"; INK C;"█"
 120 NEXT C
 130 PRINT ''" LOS COLORES PUEDE
N SER IGUALES"
 140 PRINT ''" PULSA UNA TECLA
PARA EMPEIAR."
 150 PAUSE 4E4
 160 CLS
 170 PRINT AT 1,5;
 160 FOR C=1 TO 6
 190 PRINT INK 0;C;">"; INK C;"█
";
 200 NEXT C
 210 PRINT '
 220 LET C(1)=INT (RND*6)+1
 230 LET Z=1
 240 LET Z=Z+1
 250 LET C(Z)=INT (RND*6)+1
 260 LET J=0
 270 LET J=J+1
 280 IF (J)=C(Z) THEN GO TO 230
 290 IF J<Z-1 THEN GO TO 270
 300 IF Z<4 THEN GO TO 240
 310 FOR G=1 TO 10: POKE 23692,-
1
 320 PRINT INK 0;" ENTRA TU CODI
GO SUPUESTO NUM.";G
 330 INPUT A
 335 FOR Q=1 TO 32: PRINT CHR$ 8
;: NEXT Q
 340 PRINT "
      "
 350 FOR Z=1 TO 4
 360 LET G(Z)=A-10*INT (A/10)
 370 LET H(Z)=G(Z)
 360 LET A=INT (A/10)
 390 NEXT Z
 400 LET B=0: LET W=0
```

```
 410 FOR Z=1 TO 4
 420 IF C(Z)<>G(Z) THEN GO TO
450
 430 LET B=B+1: BEEP .2,B*15
 440 LET G(Z)=0
 450 NEXT Z
 460 FOR Z=1 TO 4
 470 IF G(Z)=0 THEN GO TO 520
 480 FOR J=1 TO 4
 490 IF C(Z)<>G(J) THEN GO TO
510
 500 LET W=W+1: BEEP .2,60-B*15
 510 NEXT J
 520 NEXT Z
 530 FOR T=4 TO 1 STEP -1
 540 PRINT INK H(T);"█ ";
 550 NEXT T
 560 PRINT INK 0;"";B;" NEGRO";
 570 IF B<>1 THEN PRINT "S";
 580 PRINT " Y ";W;" BLANCO";
 590 IF W<>1 THEN PRINT "S"
 600 IF W=1 THEN PRINT
 610 IF B=4 THEN PRINT : PRINT "
ACERTASTE EN EL INTENTO NUM.";G
 630 IF B<>4 THEN NEXT G
 650 PRINT ''"EL CODIGO ERA ";
 660 FOR H=1 TO 100: NEXT H
 670 FOR T=4 TO 1 STEP -1
 680 FOR H=1 TO 50: NEXT H
 690 BEEP .2,T*10: PRINT INK C(T
);"█ ";
 700 NEXT T
 710 FOR H=1 TO 60: BEEP .01,H:
NEXT H
 720 POKE 23692,-1
 730 PRINT ''"DESEA OTRO JUEGO?"
 735 PRINT TAB 8;"PULSAR S/N"
 740 LET A$=INKEY$: IF INKEY$=""
 THEN GO TO 740
 750 IF CODE A$<>CODE "N" THEN R
UN
 760 CLS
 770 PRINT '' INK RND*6;TAB RND*
15;"BIEN, ADIOS POR AHORA!"
 780 POKE 23692,-1
 790 FOR H=1 TO 25
 800 NEXT H
 810 GO TO 770
```

```
    ESTOY PENSANDO EN UN CODIGO
DE 4 COLORES. TIENES 10 INTENTOS
PARA ACERTARLO. ESCOJO ENTRE
ESTOS COLORES
        1 >>█
         2 >>█
          3 >>█
           4 >>█
            5 >>█
             6 >>█
```

```
LOS COLORES PUEDEN SER IGUALES

PULSA UNA TECLA  PARA EMPEZAR.
```

Línea 20 (POKE 23609,100) cambia el ritmo del "clic" cuando se pulsa una tecla de sonido para actuar como una realimentación positiva cuando se hace tal pulsación. Tenemos tendencia a utilizar esta instrucción siempre y la encontramos muy útil para la programación. La línea 60 establece la tinta y el margen (INK,BORDER) negros (0) y el fondo (PAPER) blanco (7). Las rutinas desde las líneas 100 a la 120 presentan los seis colores (poniendo un cuadrado en cada color) en diagonal, con números próximos a los colores a que se refieren. La línea 150 espera hasta que se pulse cualquier tecla antes de continuar.

La rutina desde la línea 220 a la 300 selecciona los colores, asegurando que los cuatro sean diferentes. La 210, mientras tanto, ha desplazado la posición de impresión una línea hacia abajo (mediante el apóstrofo de la tecla 7, al que se accede pulsando simultáneamente el "SHIFT" rojo), y las líneas 180 a 200 han presentado los seis colores en línea en la parte superior de la pantalla con los números correspondientes.

La línea 310 empieza el bucle para proporcionar las diez opciones a adivinar. La segunda mitad de la línea 310 (POKE 23692,-1) asegura que si la pantalla se llena, correrá automáticamente sin solicitar una respuesta a la pregunta "scroll?" que, en otro caso, se encontraría con frecuencia en la parte inferior de la pantalla. Junto con el sonido de pulsación de tecla también usamos frecuentemente este desplazamiento ("scroll automático") POKE.

La línea 320 pide la solución supuesta, y una vez se ha introducido 330, utiliza el retroceso (CHR$8) 32 veces para volver a la línea que solicita la entrada de la hipótesis supuesta. La línea 320 la sobreimpresiona con espacios en blanco. Esto significa que se borra la línea INTRODUZCA HIPOTESIS 2 pero no ocurre lo mismo con las hipótesis anteriores (y el código de colores de la parte superior de la pantalla), de forma que se pueden mirar tales hipótesis como ayuda para encontrar la solución. Se introduce la solución supuesta, entrando un número de cuatro cifras, haciendo uso del código de color que tenemos en la parte superior de la pantalla. Es decir, si queremos introducir el color azul, pulsaremos el 1.

La rutina de las líneas 350 a 390 disocia el número que hemos aplicado en cuatro dígitos separados, las variables para negros (B) y blancas (W) se ponen a cero en la línea 400, y entonces la solución su-

puesta introducida se compara con el código de cuatro dígitos que ha establecido el ordenador, produciéndose unos pequeños pitidos para los "blancos" y "negros" a medida que los encuentra. Si usted ha acertado, el programa lo dice. En caso negativo, y si no ha consumido las diez oportunidades, se indicarán los dígitos del color correcto en la adecuada posición (negros) y el color acertado en posición equivocada (blancos), a la vez que se le brinda otra oportunidad.

Ya sabrá que puede utilizar PRINT AT 4,7; "TEST" para presentar esta última palabra cuatro líneas abajo y para empezar siete espacios a la derecha. El carácter de control CHR$ 22 se comporta igual que PRINT AT pero con una diferencia. Para conseguir el mismo resultado que en la instrucción PRINT AT 4,7; "TEST" es necesario introducir PRINT CHR$ 22 + CHR$ 4 + CHR$ 7; "TEST". Sin embargo, debido a que el Spectrum permite la concatenación (adición de cadenas alfanuméricas), se pueden añadir todos estos "CHR$" para formar una sola cadena. Esto puede ser muy útil si se desea especificar una determinada localización PRINT AT varias veces en un programa. Ejecute el siguiente programa y verá cómo actúa:

```
  10 LET A$=CHR$ 22+CHR$ 4+CHR$
7
  20 PRINT A$;"TEST"
```

Puede simularse el funcionamiento de la instrucción TAB haciendo preceder al CHR$n— donde n es el número de espacios (más uno) donde se desea iniciar la presentación —, con CHR$ 23. Ejecute el siguiente programa para ver este planteamiento en acción. Sin embargo, dado que CHR$ 23 realmente espera estar seguido por dos números (n y m que tienen el mismo efecto que PRINT TAB n + 256*m), puede hacerse preceder la información dentro de las comillas con un espacio vacío o una letra de simulación (X en nuestro ejemplo), que no será presentada. Ejecute el siguiente programa y verá que en lugar de aparecer XTEST en la parte inferior derecha de la pantalla, solamente se verá TEST.

```
10 LET A$=CHR$ 23+CHR$ 4
20 PRINT A$;"XTEST"
30 GO TO 20
```

Al principio de esta sección se trató de ocho colores, y se expuso cómo podían utilizarse con la información que se presentaba (INK), con respecto al fondo (PAPER) y al margen (BORDER). La informa-

ción presentada puede modificarse con la utilización de dos instrucciones adicionales. BRIGHT (brillo) y FLASH (destelleo). La rutina siguiente muestra esta acción. Introdúzcala y ejecútela, volviendo después al libro para tratar brevemente de estas dos nuevas instrucciones.

```
  40 PRINT INK 4;"NORMAL [redacted]"
  45 PRINT BRIGHT 1; INK 4;"BRIL
LO [redacted]"
  50 PRINT INK 4; PAPER 2;"NORMA
L [redacted]"
  55 PRINT BRIGHT 1; INK 4; PAPE
R 2;"BRILLO [redacted]"
  60 PRINT FLASH 1; INK 4;"DESTE
LLO [redacted]"
  65 PRINT BRIGHT 1; FLASH 1; IN
K 4;"BRILLO [redacted]"
  70 PRINT FLASH 1; PAPER 2; INK
 4;"DESTELLO [redacted]"
  75 PRINT BRIGHT 1; FLASH 1; PA
PER 2; INK 4;"BRILLO [redacted]"
```

Aunque el efecto del destelleo (FLASHING) es imposible que pase desapercibido, será preciso mirar con un poco más de detenimiento para ver el de brillo (BRIGHT). Una vez que haya ejecutado este programa, mire la palabra BRILLO (BRIGHT), justo debajo de NORMAL, cerca de la parte superior de la pantalla. Verá que hay una tonalidad diferente del verde. El blanco sobre el verde (la sexta línea en la parte inferior de la pantalla) muestra el efecto de BRILLO más claramente. Compare la claridad de la palabra BRILLO aquí con la palabra destelleo, justo encima de ella. En las palabras sin destelleo, con los colores verde o rojo (una extraordinaria combinación), apreciará que aquellas a las que se aplica "brillo" son algo más fáciles de leer que las "normales".

Aunque ya se han explicado los números del cero al siete que se aplican a los colores de tinta (INK), fondo (PAPER) y margen (BORDER), hay otros que pueden utilizarse. Usando el 8 (como en "PAPER 8"), se expresa que cualquier cosa que se imprima en ese punto no variará de color. Esto no es particularmente útil en la programación ordinaria pero el número 9 puede ser muy efectivo.

Este número significa contraste y asegura que si se está representando sobre un fondo claro la presentación se hará en negro, y se hará en blanco en caso de fondo oscuro, algo parecido al cambio de color en las instrucciones INPUT que depende del color del margen (BORDER). El siguiente programa muestra esto en acción, presentan-

do letras del alfabeto escogidas aleatoriamente e igualmente situadas al azar, contra un color del fondo (PAPER) arbitrariamente seleccionado. Ejecute el programa durante un rato para ver esto y después vuelva al libro para ver nuestras nuevas instrucciones gráficas útiles.

```
 60 PAPER RND*6
 70 CLS
 80 INK 9
100 FOR G=1 TO 32
110 PRINT AT RND*20,RND*30;CHR$
(65+INT (RND*26));
120 NEXT G
130 GO TO 60
140 PRINT AT RND*20,RND*30; OVE
R 1;CHR$ (65+INT (RND*26));
```

La palabra OVER es muy útil y puede producir efectos muy curiosos. Habrá observado al final del programa anterior una línea aparentemente innecesaria. Mediante el control de edición (tecla EDIT) pase el contenido de la línea 140 al lugar de la 110, y cambie el 32, al final de línea 100, por 300. Advertirá, de vez en cuando, que hay letras que aparecen sobre otras que han sido presentadas antes. La instrucción OVER hace que las nuevas marcaciones no eliminen las que están debajo sino que sencillamente las complementan, sustrayendo unas de las otras para hacer una nueva configuración. Esto nos permite componer algunos caracteres propios. Aplique y ejecute el nuevo programa para crear formas. Es muy difícil predecir el efecto de la adición de varias letras de esta manera. Por ejemplo, una "o" minúscula con una "w" del mismo tipo producen lo que parece una "T" mayúscula.

```
10 OVER 1
20 FOR G=1 TO 16
30 INPUT "ENTRA UNA LETRA";A$
40 INPUT "ENTRA OTRA LETRA";B$
50 PRINT AT G,G;A$;CHR$ 8;B$
60 NEXT G
```

Recordará que tratamos anteriormente de la forma en que podían utilizarse las instrucciones CHR$ 22 y CHR$ 23 para sustituir a PRINT AT y TAB, y la posibilidad de concatenar aquellos (sumarlos) para constituir una sola cadena. Lo mismo puede hacerse con las otras instrucciones. Los caracteres de control y las instrucciones a las que substituyen son: CHR$ 16 - INK; CHR$ 17 - PAPER;CHR$ 18 - FLASH;

CHR$ 19 - BRIGHT; CHR$ 20 - INVERSE; CHR$ 21 - OVER. Estas instrucciones van seguidas del número que corresponde al color o función requeridos. Como decíamos, pueden sumarse entre sí, tal cual se ve en el programa siguiente:

```
  20 INPUT PAPER 6; INK 1;"ENTRA
 UN COLOR PARA LA TINTA";A
  30 INPUT INK 2;"ENTRA UN COLOR
 PARA EL FONDO";B
  40 INPUT FLASH 1; BRIGHT 1; IN
K 4; PAPER 2;"ENTRA UNA PALABRA"
;A$
  50 LET A$=CHR$ 16+CHR$ A+CHR$
17+CHR$ B+A$
  60 PRINT AT 10,10;A$
```

La línea 60 de este programa podría también, por supuesto, añadirse a la cadena alfanumérica (string) A$. Quizás le gustaría tratar de hacerlo como ejercicio.

En el manual se explican caracteres de control adicionales. En el manual hay una tabla que proporciona una descripción completa de los diversos efectos disponibles en la hilera superior del teclado.

Si desea ver lo efectivo que puede ser el color, incluso en un programa sencillo, aplique y ejecute la rutina siguiente. Si los pitidos le molestan, borre las líneas 90 y 100. Si desea que la construcción del gráfico se haga más rápida, cambie el 7 al final de la línea 40 por un 6 para que no se representen los cuadrados blancos.

```
  10 PAPER 7: BORDER 0: CLS
  20 LET A=RND*10
  30 LET B=RND*16
  40 LET Z=RND*7
  50 PRINT AT A,B; INK Z;"█"
  60 PRINT AT 21-A,B; INK Z;"█"
  70 PRINT AT 21-A,31-B; INK Z;"
█"
  80 PRINT AT A,31-B; INK Z;"█"
  90 IF RND>RND THEN GO TO 20
 100 BEEP RND/30,RND*60-RND*60
 110 GO TO 20
```

Cuando haya disfrutado un rato con este programa, modifíquelo como sigue:

```
 10 PAPER  7. BORDER 0: CLS
 20 LET A=RND*10
 25 LET F=RND
 30 LET B=RND*16
 40 LET Z=RND*6
 50 PRINT AT A,B; FLASH F; BRIG
HT 1;EXP Z;"■"
 60 PRINT AT 21-A,B; FLASH F; B
RIGHT 1; INK Z;"■"
 70 PRINT AT 21-A,31-B; FLASH F
; BRIGHT 1; INK Z;"■"
 80 PRINT AT A,31-B; FLASH F; B
RIGHT 1; INK Z;"■"
 90 IF RND>RND THEN GO TO 20
100 BEEP RND/30,RND*60-RND*60
110 GO TO 20
```

Observará que se ha aumentado el brillo de cada cuadradito y que se ha añadido un destelleo (FLASH) a cada circuito del programa. Tanto cuando la intrucción BRIGHT como cuando la FLASH encuentran un 1, se activan; es decir, aumenta el brillo del cuadrado o se inicia el destelleo. Si tales instrucciones encuentran un cero, no se activan. FLASH y BRIGHT, como varios otros, no convierten en enteros los números aleatorios producidos por el ordenador (función INT por entero), sino que los redondean al más próximo de donde el entero (INT) de un número positivo siempre es el más próximo por *debajo* del número más fracción (0.5). En consecuencia, el efecto de la línea 25 es activar el FLASH en algunos bucles del programa y desactivarlo en otros. Podrá apreciar esto cambiando el RND de la línea 25 por 1 y ejecutando el programa durante un rato. Después lo hace cero y vuelve a observarlo. Finalmente, es posible que le guste modificar el programa para convertirlo en el siguiente que denominamos "Sopa de Letras Griegas", nombre que comprenderá cuando lo haya visto. Esta versión final resume muchos de los puntos tratados hasta ahora en esta sección.

```
  7 REM SOPA DE LETRAS GRIEGAS
 10 PAPER 7: BORDER 0: CLS
 20 LET A=RND*10
 25 OVER 1
 30 LET B=RND*16
 40 LET Z=RND*7
 45 LET A$=CHR$ (65+RND*25)
 50 PRINT AT A,B; BRIGHT 1; INK
Z;A$
 60 PRINT AT 21-A,B;; BRIGHT 1;
INK Z;A$
```

```
  70 PRINT AT 21-A,31-B; BRIGHT
1; INK Z;A$
  80 PRINT AT A,31-B; BRIGHT 1;
INK Z;A$
  90 GO TO 20
```

La realización de gráficos con gran resolución

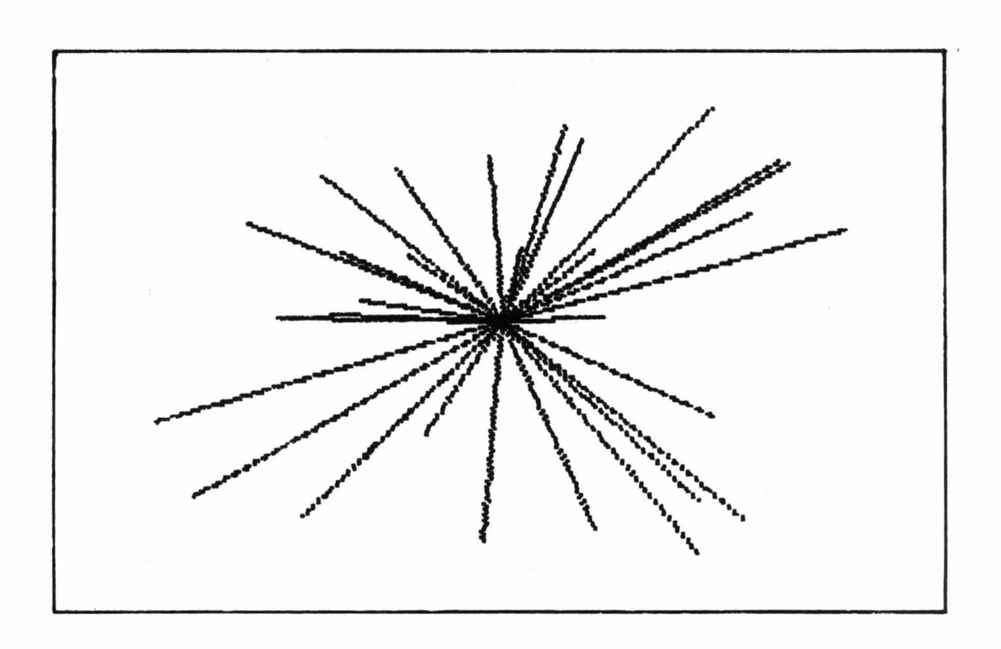
El PLOT (Trazador) permite gráficos con gran resolución, como puede observarse ejecutando los programas "Galaxia" y "Seno sólido". Una vez visto este último, se habrá dado cuenta que mientras la resolución de puntos viene en función del número de los posibles en las dos dimensiones de la pantalla (256 × 192), la resolución del color es sólo de 32 × 22. A pesar de esta merma, pueden conseguirse, sin embargo, dibujos de alta resolución con una definición muy efectiva.

```
 10 REM GALAXIA
 20 PAPER 0: BORDER 0: CLS
 30 LET C=255: LET D=175
 40 INK RND*7
 50 LET A=C*RND
 60 LET B=D*RND
 70 PLOT A,B: PLOT A,D-B
 80 PLOT C-A,B: PLOT C-A,D-B
 90 IF RND>.5 THEN GO TO 60
 95 INK RND*7
100 GO TO 50
```

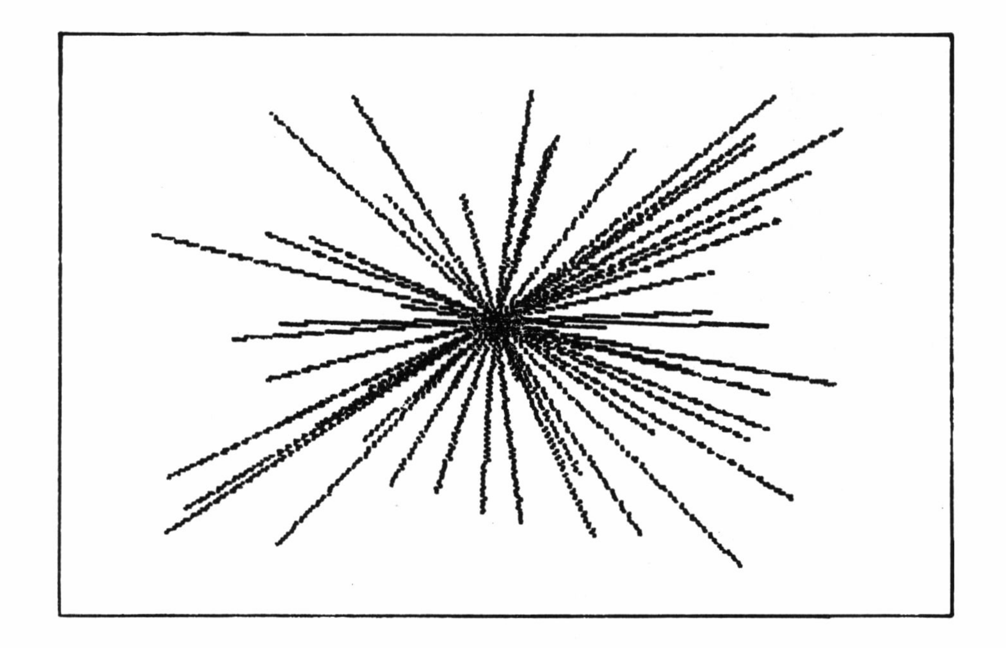
```
  10 REM FUNCION SENO EN COLOR
  20 REM ©COLIN HUGHES,
        HARTNELL   1982
  30 BORDER 2: CLS
  40 FOR X=0 TO 63 STEP .5
  50 LET Y=20*SIN (X/32*PI)
  60 IF Y=0 THEN GO TO 100
  70 FOR N=0 TO Y STEP SGN Y/4
  80 PLOT INK RND*6;X*3+30,3*(N+
30)
  90 NEXT N
 100 BEEP .1,X: NEXT X
```

El trazado de dibujos

Es posible apreciar la eficacia con la que se pueden lograr gráficos, mediante el siguiente programa, "Vidrio Roto", que utiliza la instrucción DRAW. El fondo de la pantalla (PAPER) se hace blanco (línea 30), el margen (BORDER) (línea 50) y colores de la tinta (INK) (línea 60) se establecen con variación aleatoria. La línea 70 se asegura que tales colores sean diferentes. Si no lo son, se escoge un nuevo color de la tinta. La pantalla se despeja por la línea 100 y se seleccionan aleatoriamente un par de coordenadas. Se marca un punto en el centro de la pantalla (línea 130) y se traza una recta desde ese punto al de coordenadas previamente escogidas. DRAW determina la longitud que tal línea ha de tener y con qué ángulo ha de ser trazada pero necesita que se le dé un punto de partida. Este punto es proporcionado en este programa mediante el uso del PLOT.



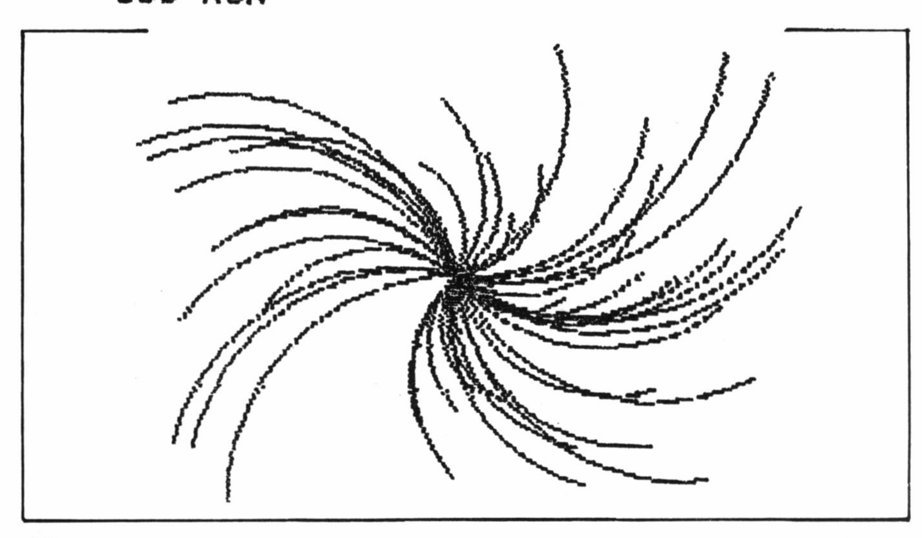
```
 10 REM VIDRIO ROTO
 20 REM © HARTNELL, RUSTON 1982
 30 PAPER 7
 50 LET A=INT (RND*8)
 60 LET B=INT (RND*7)
 70 IF A=B THEN GO TO 50
 80 BORDER A
 90 INK B
100 CLS
110 LET C=INT (RND*256)-128
120 LET D=INT (RND*172)-85
130 PLOT 128,86
140 DRAW C,D
145 BEEP .01,RND*100-50
150 IF RND>0.02 THEN GO TO 110
160 RUN
```



DRAW traza líneas cuando va seguido de dos números que son las coordenadas PLOT del punto final de la línea. Si se añade un tercer número, DRAW trazará parte de un círculo, siendo este tercer número el ángulo de giro. El programa "curvas rotas", que es igual al "vidrio roto", excepto el final de la línea 140, traza una especie de versión de barrido del "Vidrio Roto" al girar la línea un ángulo en radianes de "pi" partido por dos.

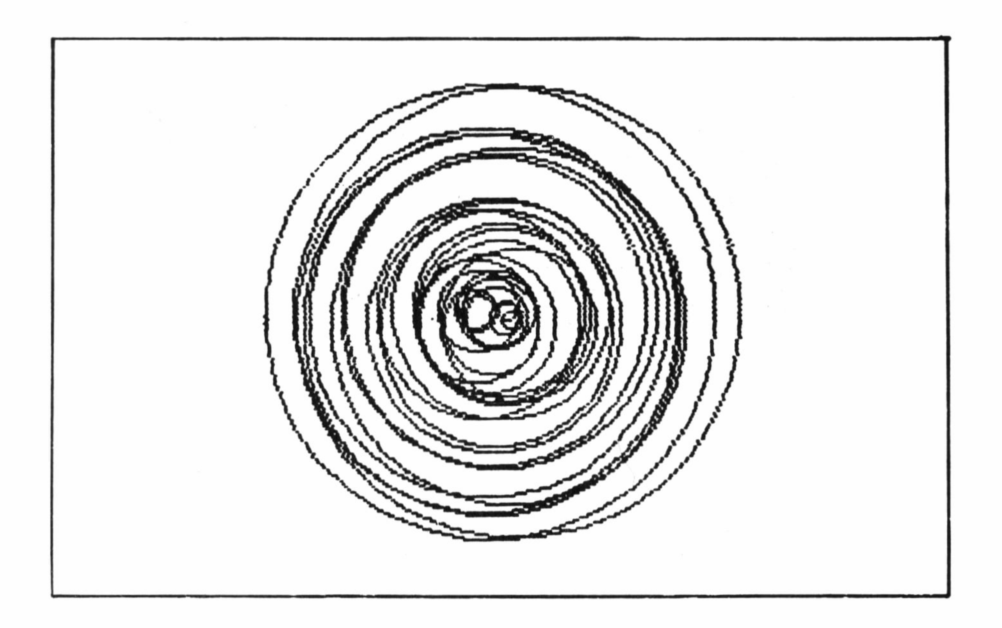
```
 10 REM CURVAS ROTAS
 20 REM © HARTNELL, RUSTON 1982
 30 PAPER 7
 50 LET A=INT (RND*8)
 60 LET B=INT (RND*7)
 70 IF A=B THEN GO TO 60
 80 BORDER A
 90 INK B
100 CLS
110 LET C=INT (RND*256)-128
120 LET D=INT (RND*172)-86
130 PLOT 128,86
140 DRAW C,D,PI/2
145 BEEP .01,RND*100-50
150 IF RND>0.015 THEN GO TO 110
160 RUN
```

```
 10 REM CURVAS ROTAS
 15 REM COLOR CAMBIANTE
 20 REM © HARTNELL, RUSTON 1982
 30 PAPER 7
 50 LET A=INT (RND*8)
 60 LET B=INT (RND*7)
 70 IF A=B THEN GO TO 60
 80 BORDER A
 90 INK B
100 CLS
110 LET C=INT (RND*256)-128
120 LET D=INT (RND*172)-86
130 PLOT 128,86
135 IF RND>0.5 THEN INK RND*6
140 DRAW C,D,PI/2
145 BEEP .01,RND*100-50
150 IF RND>0.015 THEN GO TO 110
160 RUN
```

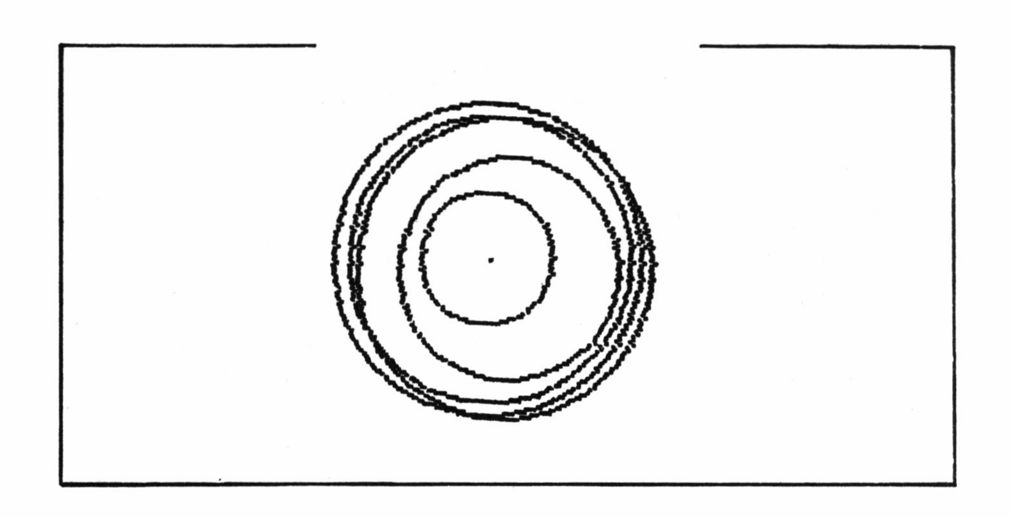


La sentencia "CIRCLE"

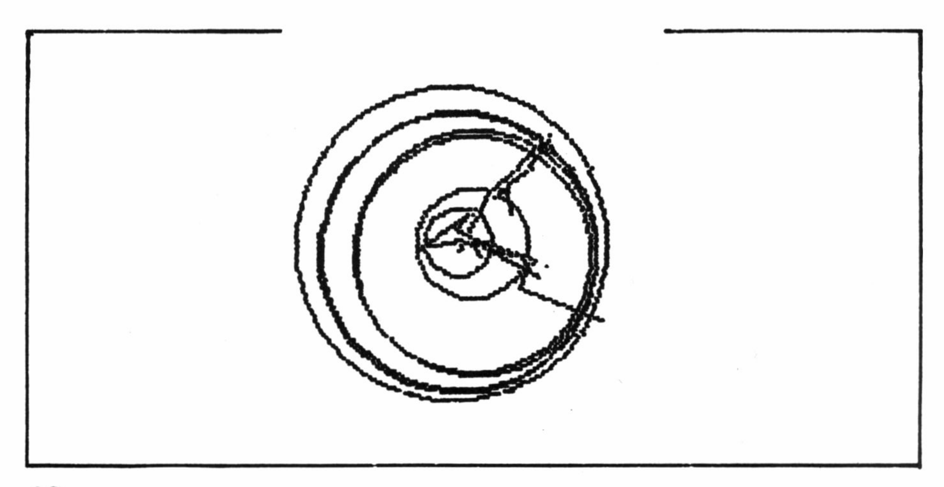
Existe otra sentencia —CIRCLE— que dibuja círculos. El programa "Visión Túnel" establece un margen (BORDER) azul pálido sobre fondo (PAPER) blanco y a continuación dibuja una serie de círculos de color aleatorio alrededor de un centro cuya posición sufre pequeños cambios también aleatorios y cuyo radio varía de igual forma. El primer número después de la sentencia CIRCLE es la abscisa del centro y el segundo la ordenada, mientras que el tercero corresponde al radio.



```
  10 REM VISION TUNEL
  15 BORDER 5: PAPER 7: CLS
  20 CIRCLE INK RND*6;128+RND*10
-RND*0,86+RND*7-RND-7,RND*65
  30 IF RND>.92 THEN CLS
  40 BEEP RND/3,RND*100-30
  50 GO TO 20
```



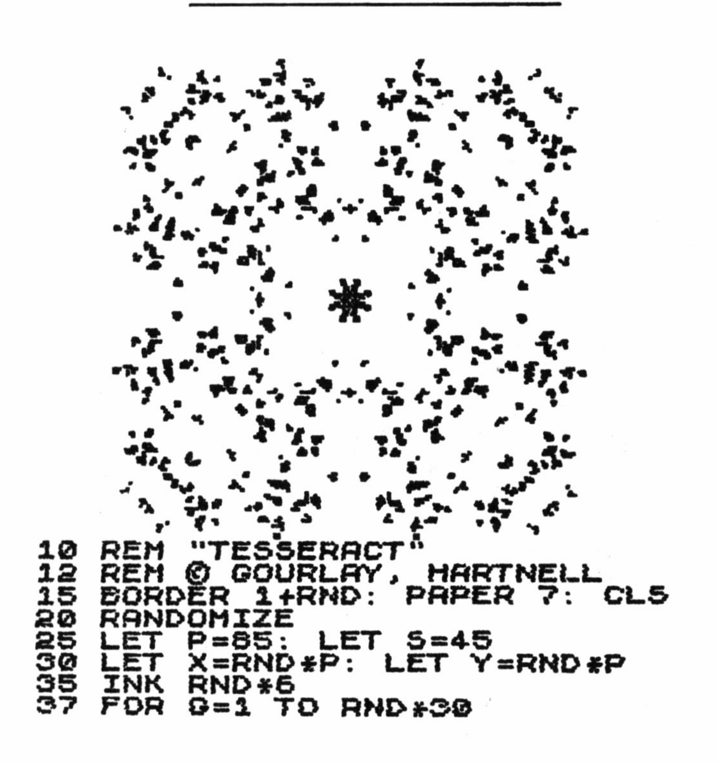
```
   10 REM VISION TUNEL
   15 REM CON MARCHA ZIGZAGUEANTE
   20 CIRCLE INK RND*6;126+RND*10
-RND*10,86+RND*7-RND-7,RND*65
   25 PLOT 128,86
   30 DRAW INK RND*6; OVER 1,32-R
ND*64,21-RND*42
   40 BEEP RND/4,RND*50-30
   50 IF RND>.33 THEN GO TO 30
   60 IF RND<.93 THEN GO TO 20
   70 RUN
```



Finalmente aquí se presentan más programas para demostrar la eficacia del Spectrum en el manejo de gráficos.

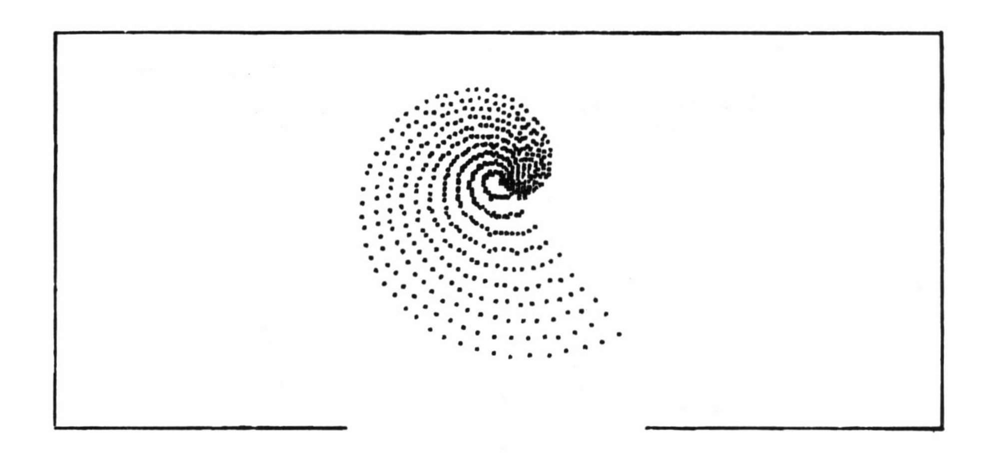
```
   5 PAPER RND*7
   7 REM BAILE DE CUADRADOS COLO
READOS"
  10 REM DEMOSTRACION DE PANTALL
A
  20 BORDER 2
  25 PAPER 7
  27 CLS
  30 PRINT INK RND*5+1;AT RND*21
,RND*31;"■"
  40 GO TO 30
```

```
  10 REM SONIDO Y VISION
  20 BORDER RND*7: PAPER RND*7:
CLS
  25 PRINT AT 10,0; INK RND*7;"■S
ONIDO Y VISION"
  30 BEEP RND*50/200,RND*50
  40 GO TO 20
```

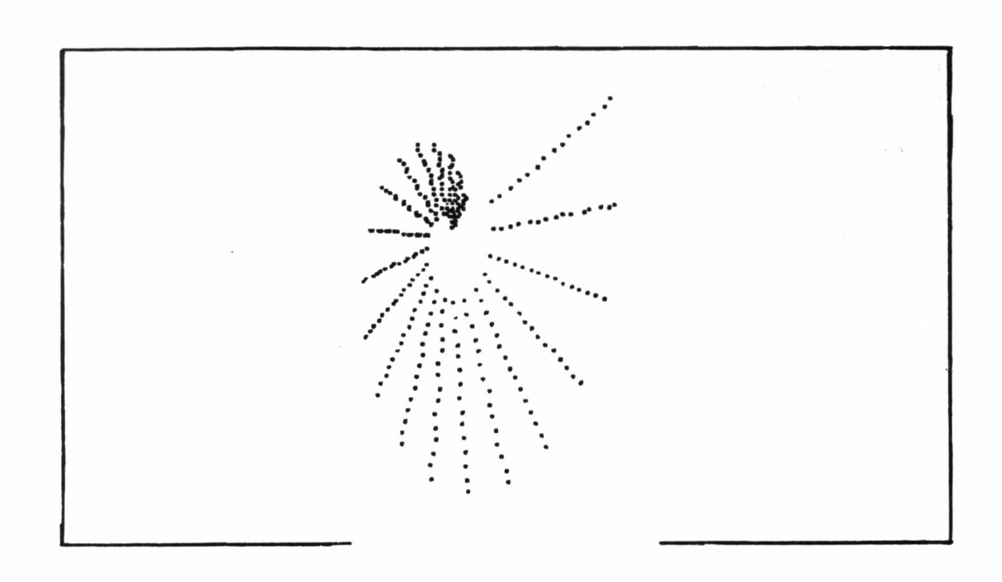


```
  10 REM "TESSERACT"
  12 REM © GOURLAY, HARTNELL
  15 BORDER 1+RND: PAPER 7: CLS
  20 RANDOMIZE
  25 LET P=85: LET S=45
  30 LET X=RND*P: LET Y=RND*P
  35 INK RND*6
  37 FOR Q=1 TO RND*30
```

```
 40 PLOT P+X+5,P+Y
 50 PLOT P+Y+5,P+X
 60 PLOT P-X+5,P+Y
 70 PLOT P+Y+5,P-X
 80 PLOT P-X+5,P-Y
 90 PLOT P-Y+5,P-X
100 PLOT P+X+5,P-Y
110 PLOT P-Y+5,P+X
120 LET X=X+RND+RND-1
130 LET Y=Y+RND+RND-1
135 NEXT G
140 IF RND<.2 OR ABS X>P OR Y>P
THEN GO TO 40
150 IF RND<.01 THEN GO TO 25
160 GO TO 30
190 RUN
```



```
  10 REM REMOLINO
  20 PAPER 7: CLS
  30 INK 0
  40 BORDER 7
  50 FOR A=1 TO 10
  60 FOR J=1 TO 7 STEP .1
  70 PLOT A*J*COS (J)+110,A*J*SI
N (J)+100
  80 NEXT J
  90 INK A/2
 100 NEXT A
```



```
   10 REM INTERLOCUTOR
   20 PAPER 7: CLS
   30 INK 0
   40 BORDER 7
   50 FOR A=5 TO 19
   60 FOR J=1 TO 7 STEP .3
   70 PLOT A*J*COS (J)/2+110,A*J*
SIN (J)+100
   80 NEXT J
  100 NEXT A
```

```
   10 REM CAMPO DE FUERZA
   15 REM (MUY LENTO)
   20 PAPER 7: CLS
   30 INK 0
   40 BORDER 7
   50 FOR A=2 TO 19
   60 FOR J=1 TO 7 STEP .01
   70 PLOT A*(J+4*SIN (J))*COS (J
)+110,A*(J+2*SIN (J))*SIN (J)+10
0
   80 NEXT J
   90 INK A/3.5
  100 NEXT A
```

El arte de utilizar las cadenas ("strings")

El programa, creado por Jeremy Ruston, hace rebotar dos pelotas alrededor de la pantalla (X,Y y L,M) y después traza líneas entre ellas. Una característica extra es que se escogen nuevas velocidades frecuentemente haciendo que las pelotas reboten en puntos distintos que no se hallan en los lados de la pantalla. Para eliminar esta peculiaridad, bórrese la línea 116.

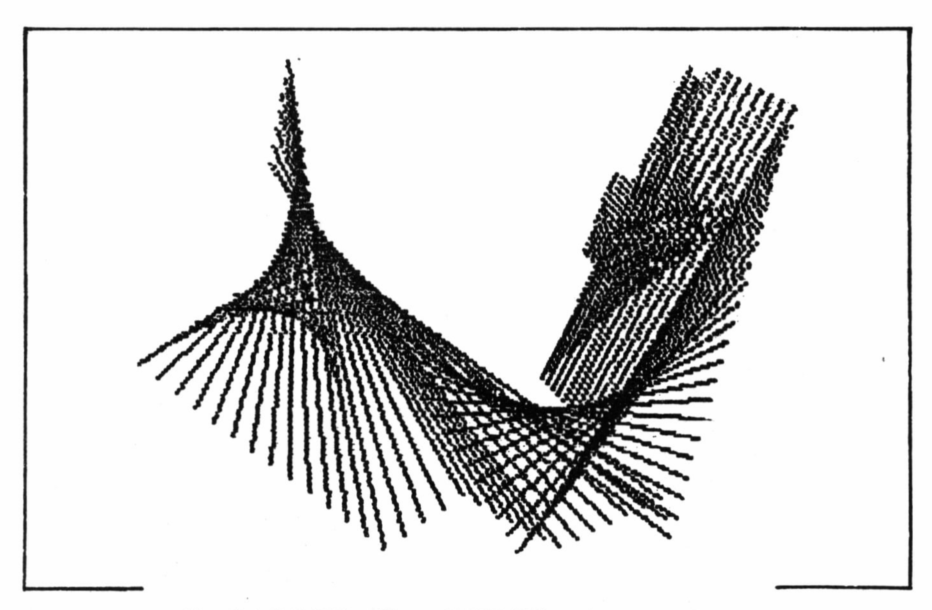
DESCRIPCION DE LAS LINEAS

5 Establece los colores negro sobre fondo blanco. Se necesita esta línea ya que el programa puede ejecutarse con distintos colores.
10 Escoge una abscisa X aleatoria para la primera pelota.
20 Escoge una ordenada Y aleatoria para la primera pelota.
30 Escoge una abscisa X aleatoria para la segunda pelota.
40 Escoge una ordenada Y aleatoria para la segunda pelota.
50 Inicia las variables requeridas para elegir las velocidades de las pelotas.
60 Define un buen generador de números aleatorios.
100 Acude a la subrutina de la línea 1000 que selecciona las velocidades aleatorias de las pelotas.
110 ¡Exhibicionismo puro!
115 La variable "num" corresponde al número de pasos que han de darse antes de elegir nuevas velocidades.
116 Si la variable "num" toma el valor cero, es el momento de seleccionar nuevas velocidades. La rutina en la línea 1000 también escoge un nuevo valor para "num".
120 Desplaza a X e Y.
130 Une X e Y con L y M.
140 Si al sumar la abscisa X de la primera pelota a su velocidad la llevase fuera de la pantalla, se invertiría el sentido del movimiento al hacerse negativa la velocidad.
150 Véase 140.
160 Véase 140.
170 Véase 140.
180 Suma la velocidad de la primera pelota a sus coordenadas.

190 Hace lo mismo para la segunda pelota.
200 Otra vez exhibicionismo...
210 Genera un número aleatorio entre 0 y 199 inclusive.
220 Si el número es 1, ejecuta el programa (RUN), despejando la pantalla y creando una nueva presentación.
230 En otro caso, traza la siguiente línea de la secuencia.
1000 Hace que la velocidad X de la primera pelota sea un número aleatorio entre −V y +V.
1010 Véase 1000.
1020 Véase 1000.
1030 Véase 1000.
1040 Escoge un valor aleatorio para la variable "num". Adviértase que no permite que tome el valor cero.
1050 Retorna al programa principal.

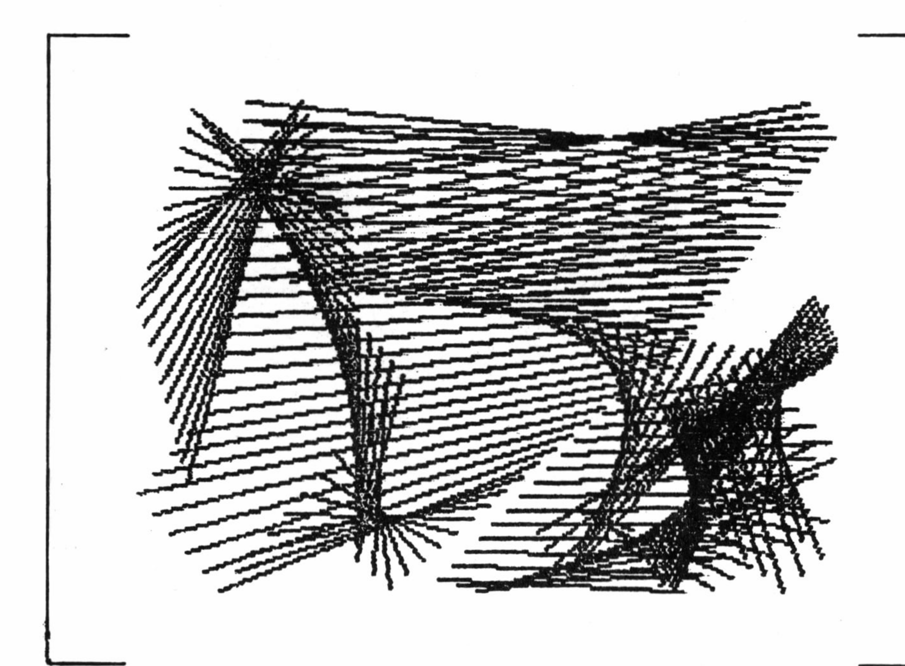
Alterando los valores de U y V de la línea 50 pueden separarse o juntarse las líneas.

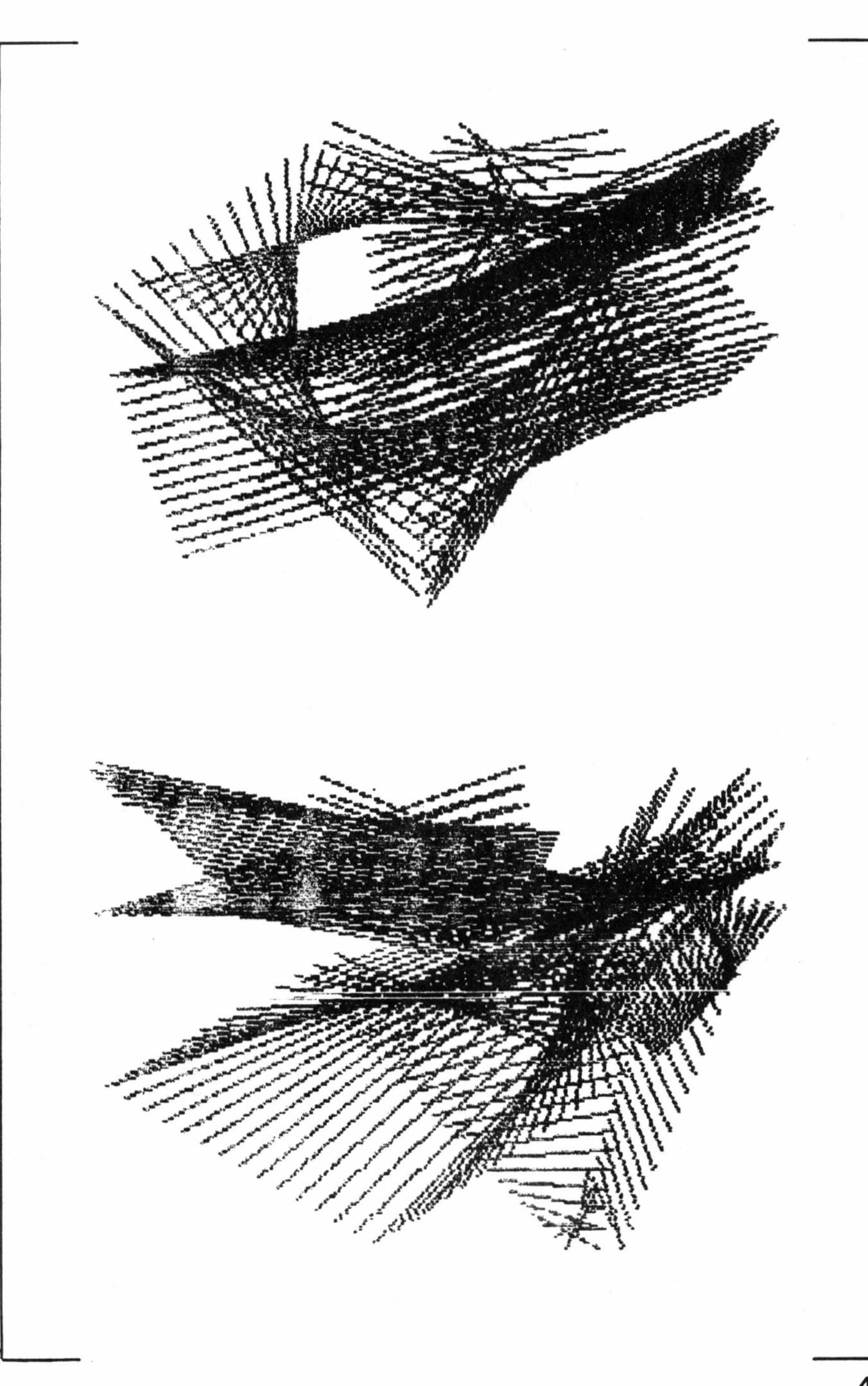
Es interesante alterar la instrucción DRAW de la línea 130 para dibujar una curva pero evítese la salida de la pantalla y procúrese no originar una señal de error.

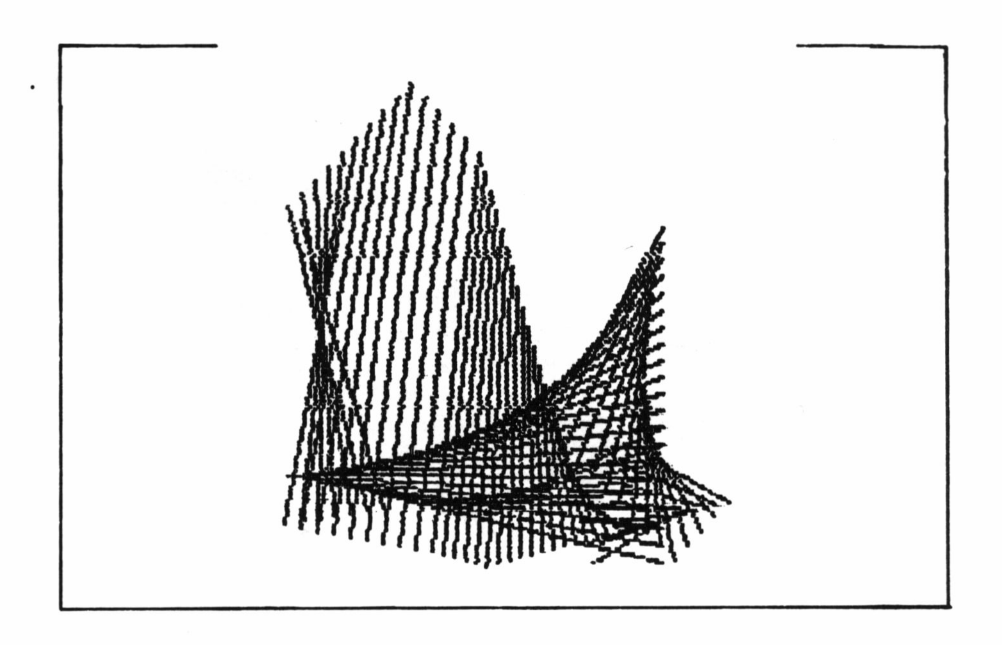


```
   5 BORDER 7: PAPER 7: CLS : IN
K 0
  10 LET X=INT (RND*256)
  20 LET Y=INT (RND*176)
  30 LET L=INT (RND*256)
```

```
  40 LET M=INT (RND*176)
  50 LET U=15: LET V=7
  60 DEF FN R(X)=INT (RND*X)
 100 GO SUB 1000
 110 REM REPITE
 115 LET NUM=NUM-1
 116 IF NUM=0 THEN GO SUB 1000
 120 PLOT X,Y
 130 DRAW L-X,M-Y
 140 IF X+A>255 OR X+A<0 THEN LE
T A=-A
 150 IF Y+B>175 OR Y+B<0 THEN LE
T B=-B
 160 IF L+C>255 OR L+C<0 THEN LE
T C=-C
 170 IF M+D>175 OR M+D<0 THEN LE
T D=-D
 180 LET X=X+A: LET Y=Y+B
 190 LET L=L+C: LET M=M+D
 200 REM HASTA QUE RESULTE FALSO
 210 LET CON=FN R(200)
 220 IF CON=1 THEN RUN
 230 GO TO 110
1000 LET A=FN R(U)-V
1010 LET B=FN R(U)-V
1020 LET C=FN R(U)-V
1030 LET D=FN R(U)-V
1040 LET NUM=FN R(20)+10
1050 RETURN
```







Figuras de moiré

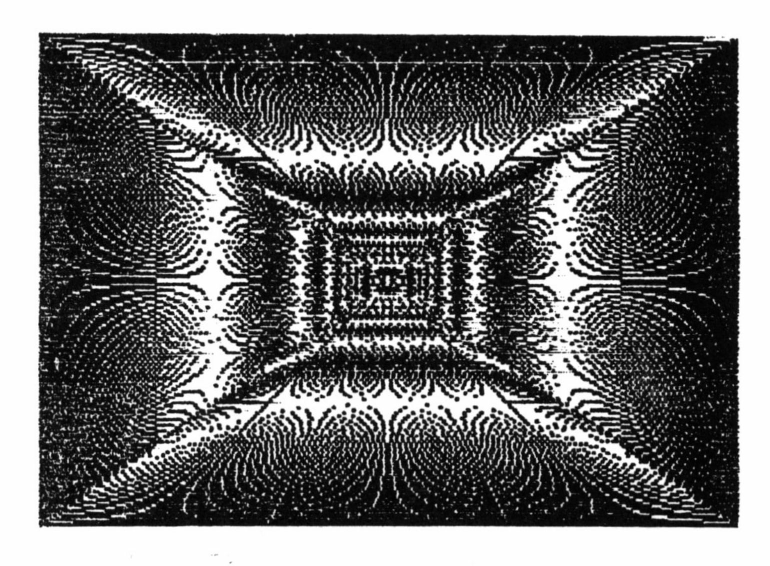
Este programa dibuja las figuras de moiré. Traza una serie de líneas desde el centro de la pantalla a cada punto de sus márgenes. Como estas líneas se trazan estando establecida la sentencia OVER, se consigue el efecto que se presenta en la figura.

Puede desearse modificar el programa para utilizar otro punto central que no sea el de la pantalla. Pruebe a eliminar las referencias OVER y aumentando el tamaño del escalón (STEP) en las dos instrucciones FOR de las líneas 10 y 50 hasta un valor próximo a cuatro. Esto proporcionará mejores figuras de moiré pero, debido a la relativamente baja resolución de la pantalla con el Spectrum, el efecto no es tan bueno como el logrado con otros ordenadores.

El color en este programa no resulta bien debido a la estrecha proximidad entre los puntos trazados.

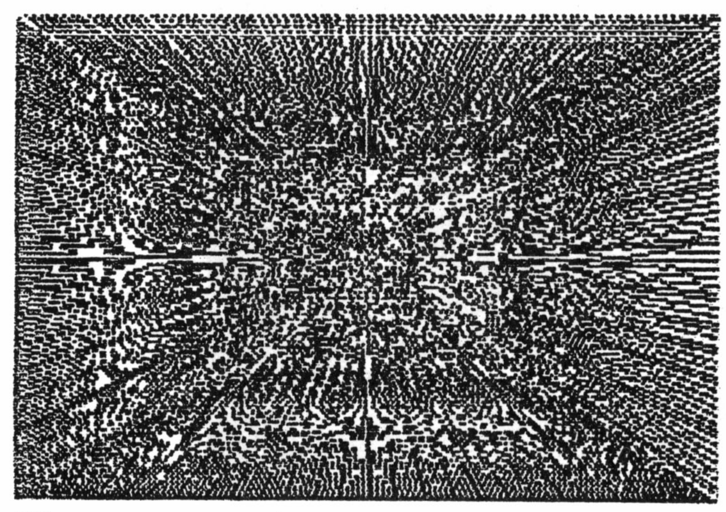
```
    5 REM TRIANGULOS GIRATORIOS
    6 REM © JEREMY RUSTON
    8 REM ************************
    9 INK 6: PAPER 0: BORDER 0: C
LS
   10 LET L=RND*256
```

```
  20 LET M=RND*176
  30 LET N=RND*256
  40 LET O=RND*176
  50 LET P=RND*256
  60 LET Q=RND*176
  70 LET A=RND*5
  80 LET B=RND*5
  90 LET C=RND*5
 100 LET D=RND*5
 110 LET E=RND*5
 120 LET F=RND*5
 130 REM REPITE
 140 PLOT L,M
 150 DRAW N-L,O-M
 160 DRAW P-N,Q-O
 170 DRAW L-P,M-Q
 171 FOR X=1 TO 50: NEXT X
 180 IF L+A>255 OR L+A<0 THEN LE
T A=-A
 190 IF M+B>175 OR M+B<0 THEN LE
T B=-B
 200 IF N+C>255 OR N+C<0 THEN LE
T C=-C
 210 IF O+D>175 OR O+D<0 THEN LE
T D=-D
 220 IF D+E>255 OR P+E<0 THEN LE
T E=-E
 230 IF Q+F>175 OR Q+F<0 THEN LE
T F=-F
 240 CLS
 250 LET L=L+A: LET M=M+B
 260 LET N=N+C: LET O=O+D
 270 LET P=P+E: LET Q=Q+F
 280 REM HASTA QUE RESULTE FALSO
 290 GO TO 130
```



```
   1 PAPER 0: INK 7: BORDER 0: C
LS
   5 REM FIGURAS DE MOIRE
   6 REM © JEREMY RUSTON
   7 REM ****************
  10 FOR X=0 TO 255
  20 PLOT X,0
  30 DRAW OVER 1;255-X*2,175
  40 NEXT X
  50 FOR Y=0 TO 175
  60 PLOT 0,Y
  70 DRAW OVER 1;255,175-Y*2
  80 NEXT Y
```

```
  5 PAPER 0: CLS : BORDER 0
 10 FOR X=0 TO 255
 20 PLOT X,0
 30 DRAW OVER 1,255-X*2,175
 40 NEXT X
 50 FOR Y=0 TO 175
 60 PLOT 0,Y
 70 DRAW OVER 1;255,175-Y*2
 80 NEXT Y: REM © J.RUSTON
 85 LET S=RND+.5
 90 FOR X=255 TO 0 STEP -S
100 PLOT X,0
110 DRAW OVER 1,255-X*2,175
120 NEXT X
130 FOR Y=0 TO 175 STEP S
140 PLOT 0,Y
150 DRAW OVER 1,255,175-Y*2
160 NEXT Y
165 PAUSE 200
170 INK RND*7: PAPER 9: CLS
180 GO TO 85
```



La instrucción POINT

Esta instrucción es con respecto a PLOT como SCREEN$ lo es en relación a PRINT AT. Es decir, se puede utilizar POINT para determinar la presencia o ausencia de un punto presentado con el enunciado PLOT en una localización determinada.

```
   10 REM BUSCA EL PUNTO
   20 FOR A=1 TO 100
   30 PLOT RND*100,RND*100
   40 NEXT A
   50 LET CON=0
   60 FOR G=1 TO 1000
   70 IF POINT (RND*100,RND*100) =
1 THEN BEEP 1,1: LET CON=CON+1:
PRINT AT 1,1;"LA TASA DE ENCUENT
ROS ";CON/G*100;" %      ";AT 3,1;
"NUMERO DE ENSAYOS: ";G
   80 NEXT G
```

Este es un programa sencillo para demostración. Produce algunos puntos con PLOT, en forma aleatoria, y después también en forma aleatoria los busca. Cada vez que el POINT encuentra un PLOT (véase la línea 70), se produce un pitido que le avisa. Puede conseguirse una tasa de aciertos de alrededor del uno por ciento.

Si el POINT, en las coordenadas X e Y, se hace igual a uno, existe un punto grabado con PLOT en esa posición. Si no ocurre así, POINT(X,Y) se hace igual a cero. Pruebe a ejecutar el citado programa con el bucle A hasta 1000 y el G hasta 10000. La tasa de aciertos será aproximadamente diez veces mayor que anteriormente (es decir, un 10 %). ¿Cuál es la razón de todo esto?

La impresora

Existen tres instrucciones que se utilizan con la impresora: LLIST, LPRINT y COPY.

LLIST Pasa todo el listado del programa a la impresora. No es posible traspasar sólo una parte del mismo. Se comporta

análogamente a la instrucción LIST, con la diferencia de que la presentación se hace en la impresora en lugar de hacerse en la pantalla.

LPRINT También actúa como PRINT pero en la impresora. Puede utilizarse en modalidad directa, como LPRINT "HOLA AMIGOS", o bien en un programa, como; 10 LPRINT "LA RESPUESTA ES";A.

COPY Copia el contenido total de la pantalla en la impresora después de haberse ejecutado un programa. Debido a que la impresora no representa colores, la copia ha de considerarse menos impresionante que la representación en pantalla. Se puede experimentar con la utilización de la instrucción INVERSE para una zona limitada para dar más calidad a la impresión.

Números aleatorios

Los números aleatorios son muy útiles para los juegos con ordenador. Examinemos su producción y utilicémoslos en algunos programas sencillos.

El ordenador permite generar dos números aleatorios con coma flotante entre cero y uno.

Ejecute el siguiente programa para crear una serie de números entre cero y uno.

```
10 PRINT RND
20 GO TO 10
```

Obtendrá una lista de números como esta:

```
.0011291504       0.6658783
.08581543         0.94125366
0.43719482        0.59406569
0.79025269        0.55688477
0.2691803         0.76586096
0.18934631        0.51463154
0.20188904        0.61291504
0.14257813        0.96907043
0.69433594        0.66031311
```

Encontrará que los enteros aleatorios son frecuentemente más útiles que estos números entre cero y uno. Para obtener los enteros aplique una instrucción como INT (RND 30) + 1. Ejecute el programa. Conseguirá una serie de números como los que produce el siguiente programa:

```
  10 REM NUMEROS ENTEROS
ALEATORIOS
  20 LET A=INT (RND*100)+1
  30 PRINT TAB 6;A
  40 GO TO 20
```

```
17
75
97
58
68
26
8
78
19
57
43
61
70
100
66
34
28
72
52
39
64
37
```

El ordenador toma el número entre paréntesis (conocido como el *argumento* de la función) y selecciona números aleatorios entre el uno y dicho número. Para obtenerlos negativos basta poner el signo menos delante de INT. Pruébelo y pase el programa otra vez para conseguir un resultado parecido a este:

```
-86      -16
-97      -78
-32      -73
-70      -79
-35      -63
-29      -8
-22      -39
-41      -64
-55      -44
-97      -10
-84      -73
```

Puede utilizarse el generador de números aleatorios para cualquier aplicación en la que sea necesario representar una actividad de este tipo, como puede ser la distribución de la mala hierba en un jardín, las nubes en el cielo o el resultado de un lanzamiento de dados. El siguiente programa es precisamente la imitación de una partida con un dado de seis caras. Aplíquelo y ejecútelo algunas veces.

```
  10 REM *TIRADA DE DADOS*
  20 PRINT "CUANTAS VECES TIRO"
  30 PRINT "EL DADO?"
  40 INPUT A
  50 CLS
  60 PRINT "RESULTADO DE TIRAR
EL DADO",;A;" VECES"
  70 FOR B=1 TO A
  80 LET C=INT (RND*6)+1
  90 PRINT ,C
 100 NEXT B
```

```
RESULTADO DE TIRAR  EL DADO
6 VECES
               3
               2
               1
               6
               4
               3
```

Corrida de toros

He aquí un sencillo juego que muestra la acción de un generador de números aleatorios. El juego en realidad no es un verdadero juego pero vale la pena practicarlo. Una vez que haya jugado unas cuantas veces vuelva al libro para tratar del programa. Resultará agradablemente sorprendido por lo mucho que ha aprendido.

Usted es un matador de toros. El animal embiste diez veces y usted selecciona un número entre el uno y el tres, mientras el toro hace lo mismo. Si los números son diferentes, se salva de la acometida pero si son iguales se termina el juego. Al final se le adjudican los puntos conseguidos.

```
  10 REM CORRIDA DE TOROS
  20 LET PUNTOS=0
  30 FOR G=1 TO 10
  40 PRINT AT 4,4; INK 2;" EL TO
RO EMBISTE"
  60 PRINT 'TAB 1; INK 1;"QUE MO
VIMIENTO HACE (DE 1 A 3)?"
  70 INPUT A
  80 IF A<1 OR A>3 THEN GO TO 70
  90 LET B=INT (RND*3)+1
 100 IF A=B THEN GO TO 220
 120 PRINT 'TAB 4; INK 4;" SALIS
TE ILESO DEL PASE "; INK 2;G
 130 BORDER RND*7
 140 PRINT ''' INK 2;"EL TORO ES
COGE ";B
 150 PRINT ' INK 4;"TU ESCOGES "
;A
 160 FOR H=1 TO 10
 170 BEEP .1,RND*50-RND*50
 180 NEXT H
 190 CLS
 200 NEXT G
 210 GO TO 230
 220 PRINT ' INK 2;"NO ERES BUEN
 TORERO"
 230 FOR T=1 TO 50: BEEP .05,RND
*50: NEXT T
 240 PRINT ' INK 2;"PUNTUACION
LOGRADA ";100*(G-1)
```

```
    EL TORO EMBISTE

 QUE MOVIMIENTO HACE (DE 1 A 3)?

    SALISTE ILESO DEL PASE 1


EL TORO ESCOGE 1

TU ESCOGES 2
```

Adviértanse los apóstrofes (') de las líneas 120, 140, 150, 220 y 240 (se aplica tal signo con la tecla del 7) que se utilizan para desplazar hacia abajo la línea de representación (la del PRINT).

Recorramos el programa línea por línea:

10 Instrucción REM de información sobre el título.

20 Hace igual a cero la variable SCORE (tanteo). Después se tratará de las variables.

30 Inicia el bucle "FOR/NEXT" para contar hasta 10. Estos bucles se estudiarán un poco después.

40 Anuncia que el toro acomete.
60 Pide al jugador que entre un número entre uno y tres.
70 Acepta el número del jugador.
80 Comprueba si el número está entre uno y tres. En caso negativo, vuelve a la línea 70 para aceptar otro dato de entrada del jugador.
90 Establece B igual al número del toro, uno igualmente escogido aleatoriamente entre uno y tres.
100 Compara el número del jugador (A) con el del toro (B) y si son iguales envía la acción a la línea 220 para comunicarle que ha fracasado como matador.
120 Le dice al jugador que se ha salvado en esta acometida.
130 Cambia el color del margen (BORDER).
140 Dice al jugador el número del toro.
150 Recuerda al jugador su número.
160-180 Establece una pequeña pausa, con música, antes de la siguiente acometida.
190 Despeja la pantalla.
200 Retorna a la siguiente acción.
210 Si el jugador se ha salvado de los diez ataques, el ordenador se remite a la impresión de la puntuación alcanzada.
220 Este es el mensaje de haber fallado en el caso de que hubiesen resultado iguales A y B en la línea 100.
230 Bucle de retardo con música.
240 Presenta la puntuación alcanzada.

Leyendo esta explicación un par de veces y mirando cuidadosamente a la línea o líneas referidas, usted puede aprender algo más sobre programación. Hay un determinado número de instrucciones específicas que estudiaremos con más detalle pero probablemente está usted empezando a comprender bastante de lo que hemos visto hasta ahora.

Variables

Habrá observado en el programa anterior que se utilizaron letras para representar números. Se asignó la letra A (en la línea 70) a un número entre uno y tres y lo mismo se hizo con B en la línea 90. Las letras A y B en este programa se denominan *variables.*

Existen dos tipos de variantes: las numéricas y las de cadena ("string") o alfanuméricas.

Casi cualquier combinación de letras y números puede utilizarse como una variable, siempre que empiece por letra y no haya signos de puntuación ni símbolos. Así TIZNAJO y D17 son nombres válidos para viariables, mientras que 2TIZNAJO y 1D7 no lo son. Las variables numéricas, letras o combinaciones de letras y números empezando por letra son de fácil utilización. Puede asignarse una variable de este tipo a cualquier número que esté dentro del alcance numérico del ordenador. El Spectrum ignora los espacios que haya en los nombres de las variables y no distingue entre letras minúsculas y mayúsculas (así, a$ es lo mismo que A$).

A propósito, como usted probablemente sabe, el ordenador utiliza la notación científica para representar grandes números con un entero seguido de hasta ocho cifras decimales seguidas de una E (para operaciones exponenciales) y la potencia de diez por la que ha de multiplicarse el número. Aplique y ejecute la siguiente demostración que muestra la variable A en acción, siendo asignada a un número que se multiplica por diez y después se presenta en la pantalla.

```
10 REM NOTACION CIENTIFICA
20 LET A=1234
30 PRINT A
40 LET A=10*A
50 GO TO 30
```

```
1234
12340
123400
1234000
12340000
1.234E+8
1.234E+9
1.234E+10
1.234E+11
1.234E+12
1.234E+13
1.234E+14
1.234E+15
1.234E+16
1.234E+17
1.234E+18
1.234E+19
1.234E+20
1.234E+21
1.234E+22
1.234E+23
1.234E+24
```

Adviértase que después de que el número tiene ocho dígitos (12340000), su presentación se hace como un número, la coma decimal, la letra E y una potencia de diez. Pruebe y anticipe la longitud del programa hasta alcanzar el máximo número posible del ordenador. Confirme su hipótesis.

Observando este listado advertimos otro par de cosas sobre las variables. Estas quedan asignadas por el solo hecho de aplicar su nombre (en este caso, A) precedido por LET y seguido del signo igual, a cuya derecha se pone el valor que hemos dado a la variable. Si decimos LET A = 99 y, a continuación, pulsamos PRINT A, aparecerá en pantalla 99. La línea 40 parece un poco extraña. El asterisco (*) es el signo de multiplicar en lenguaje BASIC. La línea 40 parece decir que A es igual a diez veces ella misma lo que, en verdadera aritmética, no es cierto. Esta es, sin embargo, la forma en que se utiliza la instrucción LET en BASIC.

Variables en cadena ("string variables")

Estas variables en cadena o alfanuméricas se representan con una letra seguida del signo del dólar ($). Aplíquese LET A$ = "HOLA", púlsese ENTER y después PRINT A$ y nuevamente ENTER. En pantalla tendremos la palabra HOLA. Puede ponerse cualquier cosa, incluyendo números, símbolos, signos de puntuación o letras, entre comillas, para constituir una variable alfanumérica. Una serie de letras y lo que sea, entre comillas, es conocido por una cadena (string).

El canto de los grillos y la temperatura

Existe, aunque parezca extraño, una correlación entre la temperatura ambiente y el número de veces que canta un grillo en un minuto. El siguiente programa, que muestra nombres de variables largas en

acción, convierte el número de cantos por minuto en un valor de temperatura en grados Fahrenheit. Ejecútese este programa algunas veces. Obsérvese que la variable "canto" se hace igual a 80 en la línea 20. Esto se convierte en la variable "temperatura" en la línea 30 y esta última variable se utiliza en la instrucción PRINT de la línea 40. La variable "canto" se incrementa en un número aleatorio comprendido entre uno y siete en la línea 60; existe un corto retraso (líneas 70 y 80) y el programa regresa a la 30 para repetir la totalidad del proceso otra vez. Se repetirá durante un tiempo muy largo (hasta que se exceda el máximo número que puede manejar el ordenador) si no lo interrumpe con la tecla "BREAK".

LA TEMPERATURA ES 62 GRADOS F
CUANDO HAY 87 CANTOS

LA TEMPERATURA ES 63 GRADOS F
CUANDO HAY 90 CANTOS

LA TEMPERATURA ES 63 GRADOS F
CUANDO HAY 92 CANTOS

LA TEMPERATURA ES 64 GRADOS F
CUANDO HAY 96 CANTOS

LA TEMPERATURA ES 65 GRADOS F
CUANDO HAY 98 CANTOS

```
   10 REM CONVERSION DE CANTOS
DEL GRILLO
   20 LET CANTO=80
   30 LET TEMPERATURA=INT ((CANTO
/4)+40.5)
   40 PRINT '"LA TEMPERATURA ES "
; INK 2;TEMPERATURA;" GRADOS F."
   50 PRINT "CUANDO SE OYEN "; IN
K 2;CANTO; INK 0;" CANTOS/MIN"
   60 LET CANTO=CANTO+INT (RND*7)
   70 FOR J=1 TO 100
   80 NEXT J
   90 POKE 23692,-1
  100 GO TO 30
```

Aunque resulta un poco largo pulsar los nombres de las variables largas presenta, sin embargo, una clara ventaja sobre la utilización de designaciones como A, B y C2. Se sabe, sin tener que mirar atrás,

lo que representa cada variable. He aquí otro programa que utiliza dos variables por su nombre para aclarar lo que se está haciendo. Ejecútelo.

```
   10 REM ** VARIABLES **
   20 LET W$="EL NUMERO ES "
   30 LET NUMERO=3
   50 PRINT W$;NUMERO
   60 PRINT '"EL CUADRADO DE ";NU
MERO
   70 PRINT TAB 5;"ES ";NUMERO*NU
MERO
   80 PRINT '"Y LA RAIZ CUADRADA"
   90 PRINT "ES ";SQR (NUMERO)
```

Resumen:

• **Variables numéricas.** — Pueden tener cualquier nombre mientras empiece por letra y no contenga signos de puntuación ni símbolos.

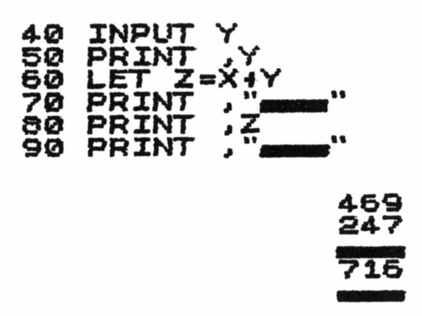
• **Variables de cadena ("string") alfanuméricas.** — Se representan por una letra seguida del signo del dólar y están constituidas por cualquier expresión encerrada entre comillas. Todas las variables se asignan mediante la instrucción LET seguida por el nombre de la variable, un signo igual y a continuación el valor que se le aplica.

La instrucción INPUT

La instrucción INPUT se utiliza para lograr información de un usuario mientras se está ejecutando un programa. El ordenador se detiene cuando llega a esta instrucción y espera una entrada de cualquier clase desde el teclado antes de continuar el proceso del programa.

Entre y ejecute el siguiente programa que muestra INPUTS numéricos en acción. El programa se detendrá en espera de que usted entre un número; a continuación se pulsa ENTER y vuelve a esperar otro número. Después de volver a pulsar ENTER, presentará la suma de los dos números.

```
10 REM ** INPUT **
20 INPUT X
30 PRINT ,X
```



```
40 INPUT Y
50 PRINT ,Y
60 LET Z=X+Y
70 PRINT ,"____"
80 PRINT ,Z
90 PRINT ,"____"
```

```
                    469
                    247
                    ____
                    716
                    ____
```

Esto está bien en la forma en que se presenta pero no se sabría qué hacer durante la ejecución a menos que se hubiera leído el libro. Hay una manera sencilla de corregir esto mediante la programación con referencias del usuario. El programa precedente puede volverse a escribir de forma que el usuario no tenga duda sobre el significado de lo que se hace.

```
10 REM ** INPUT **
20 INPUT "DAME UN NUMERO ";X
30 PRINT ,X
40 INPUT "DAME OTRO ";Y
50 PRINT ,Y
60 LET Z=X+Y
70 PRINT ,"____"
80 PRINT ,Z
90 PRINT ,"____"
```

Al ejecutarlo se observa que el ordenador presenta las palabras incluidas entre las comillas y a continuación espera la entrada del dato.

Obsérvese que muchas de las instrucciones utilizadas para controlar el PRINT pueden usarse con INPUT, como se ve en el siguiente programa.

Combate

```
 5 REM ** COMBATE **
 6 POKE 23609,100
10 LET TANTEO=0
15 FOR J=1 TO 20
```

```
  20 PRINT AT 1,8; INK J/3;"JUGA
DA NUM. ";J
  30 INPUT INK J/3;" ENTRA UN NU
MERO DEL 1 AL 10 ";A
  40 IF A<1 OR A>10 THEN GO TO
30
  50 PRINT AT 10,0; INK 3;"TU NU
MERO ES ";A;AT 12,6; INK 1;" EL
TANTEO ES ";TANTEO
  60 FOR G=1 TO 4
  70 LET B=INT (RND*10)+1
  80 PRINT AT 3,3; INK 2;B;" "
  85 FOR M=1 TO 10: BEEP .01,3.5
*M: NEXT M
  90 IF B=A THEN GO TO 110
 100 NEXT G
 110 IF A=B THEN LET TANTEO= TAN
TEO+1: PRINT AT 14,6; INK 6;"BIE
N!"
 140 IF A<>B THEN PRINT AT 14,8;
 INK 0; FLASH 1;"MALA SUERTE"
 150 PRINT AT 12,6; INK 2;"  EL
TANTEO ES ";TANTEO
 160 IF TANTEO=5 THEN GO TO 250
 170 FOR T=1 TO 20
 180 BEEP .01,2*T: BEEP .01,20-T
: NEXT T
 190 CLS
 200 NEXT J
 210 PRINT BRIGHT 1; INK 2;" EL
JUEGO HA TERMINADO"
 220 PRINT FLASH 1; BRIGHT 1;
INK 2;"Y SOLO HAS LOGRADO EL TAN
TEO DE ";TANTEO
 230 PRINT ' INK 1;"TU MEDIA ES
DEL ";TANTEO/.05;" POR CIENTO"
 240 STOP
 250 PRINT INK RND*6;  FLASH 1;
TAB 4;"LO HICISTE  "
 260 FOR M=1 TO 4: BEEP .01,RND*
50: PAUSE 3: NEXT M
 270 PRINT INK RND*6; FLASH 1;TA
B 8;"HAS GANADO!!"
 280 POKE 23692,-1
 290 GO TO 250
```

```
                JUGADA NUM. 1

      4


      TU NUMERO ES 5

                EL TANTEO ES 0

                MALA SUERTE
```

En COMBATE usted selecciona un número entre el uno y el diez y el ordenador escoge hasta cuatro comprendidos en el mismo intervalo. Si uno de ellos es igual al suyo, su tanteo se incrementa en una unidad. Al conseguir cinco puntos de veinte jugadas, usted gana. En caso contrario, se pierde y el ordenador presenta el porcentaje logrado. Una vez que haya jugado con el programa vuelva al libro para revisarlo línea por línea. Aunque se trata de un juego bastante trivial, su ejecución y las explicaciones que siguen aumentarán sus conocimientos sobre varios aspectos del BASIC y, por supuesto, le permitirá ver INPUT en acción.

5 Título.
6 Ofrece un sonido musical al presionar las teclas.
10 Pone la variable tanteo a cero.
15 Inicia la acción del ciclo FOR/NEXT principal para contar las jugadas.
20 Apunta el número de la jugada.
30 Acepta la entrada para la variable A, utilizando j para establecer el color.
40 Comprueba que la entrada es aceptable.
50 Presenta el número escogido y el tanteo. Obsérvese que los enunciados INPUT pueden encadenarse de esta manera, con punto y coma, y con la utilización de AT y TAB.
60-100 Genera hasta cuatro números. Después de la creación de cada número (línea 70), cada uno se presenta en pantalla (línea 80), produce un sonido (línea 85) y lo compara con el del jugador (línea 90).
110 El tanteo es incrementado en uno si la suposición fue correcta y en la pantalla aparece BIEN!
140 Presenta MALA SUERTE si la hipótesis es incorrecta.
150 Vuelve a presentar el tanteo.
170-180 Breve pausa con música (beeps) antes del siguiente movimiento.
190 Despeja la pantalla.
200 Fin del bucle FOR/NEXT principal.
210-240 Fin del juego, si pierde.
250-290 Fin del juego, si gana.

Interés compuesto

El siguiente programa muestra la instrucción INPUT nuevamente en acción, y también la utilización de nombres explícitos para las variables, lo que permite entender mejor lo que se está haciendo. Es posible que desee conservar este programa en una cassette ya que tiene un cierto grado de aplicación práctica.

```
  10 REM INTERES SIMPLE
  20 REM Y COMPUESTO
  30 INPUT INK 1;TAB 6;"CAPITAL?
 ";CAPITAL
  40 INPUT INK 2;TAB 4;"  INTERE
SES? ";INTERESES
  60 INPUT INK 4;TAB 4;"CUANTAS 
ANUALIDADES? ";ANUALIDADES
  80 PRINT INK 2;"A/O";TAB 7;"SI
MPLE";TAB 17;"COMPUESTO";TAB 27;
"DIF"
  90 POKE 23692,-1
 100 PRINT INK 1;"▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚
▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚"
 110 FOR M=1 TO ANUALIDADES
 120 LET SIMPLE=CAPITAL+M*CAPITA
L*(INTERESES/100)
 130 LET COMPUESTO=INT (100*CAPI
TAL*(1+INTERESES/100)↑M)/100
 140 LET DIF=INT (100*(COMPUESTO
-SIMPLE+.005))/100
 160 PRINT M;TAB 7;SIMPLE;TAB 17
;COMPUESTO;TAB 27;DIF
 170 NEXT M
```

```
A/O    SIMPLE    COMPUESTO DIF
▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚▚
1      108.25    108.25    0
2      116.5     117.18    0.68
3      124.75    126.84    2.09
4      133       137.31    4.31
5      141.25    148.64    7.39
6      149.5     160.9     11.4
7      157.75    174.17    16.42
8      166       188.54    22.54
9      174.25    204.1     29.85
10     182.5     220.94    38.44
11     190.75    239.17    48.42
12     199       258.9     59.9
```

Este programa calcula el interés simple y el compuesto para el capital y rédito que usted determine, durante el número de años que decida. El ejemplo está preparado para un capital de 100 dólares, un rédito del 8,25 %, durante 12 años.

Para detener un programa durante una entrada alfanumérica (la instrucción BREAK no actúa con el INPUT) utilice el cursor izquierdo (SHIFT,5) o el DELETE (SHIFT,0) para hacer que el cursor salga de las comillas, pulsando a continuación STOP(SHIFT,A) seguido de NEWLINE. Si se trata de un INPUT sin comillas basta pulsar STOP (SHIFT,A) y después ENTER. En ambos casos, el programa se detiene con un mensaje 9.

Es útil rechazar los INPUTS no válidos *antes* de que afecten a la marcha del programa.

Si se invita al usuario a hacer una nueva prueba, analícese su respuesta como sigue:

```
555 INPUT "DESEAS OTRA PRUEBA?
";R$
556 IF R$(1)="S" THEN RUN
```

Existe una ley en alguna parte que dice que el usuario responderá pulsando sólo ENTER, lo que nos deja con INPUT nulo. En consecuencia, no hay nada como un R$(1), no existe, como le dirá muy rápidamente el ordenador en la forma de un mensaje de error.

He aquí otro método para impedir esto:

```
550 DIM R$(1)
555 INPUT "DESEAS OTRA PRUEBA?
";R$
556 IF R$(1)="S" THEN RUN
```

Debido a que R$ ha sido previamente dimensionado con DIM ha de tener un carácter, no importa lo que se entre. Si únicamente se pulsa ENTER, entonces R$ tendrá un espacio ya que esto es lo que se aplicó a R$ después de DIM y un INPUT nulo no lo cambiará. Si INPUT (la entrada) tiene una longitud de varios caracteres, sólo existe espacio en R$ para el primero. Si esta letra es "S", el programa pasará para una nueva prueba. Este método tiene la ventaja de que si se entra una respuesta muy larga, como: "SI, POR FAVOR, AMABLE ORDENADOR, ME GUSTARIA MUCHISIMO PROBAR OTRA VEZ SU GRAN PROGRAMA" (¡muy poco probable!), no hay necesidad de albergarla entera en la memoria. También es muy útil si se hace uso del "GO TO" o no se hace nada que descarte (CLEAR) las variables, almacenando así innecesariamente toda la respuesta. El se-

gundo método es más convencional y utiliza una línea menos de programa que la rutina previa, aunque acoja innecesariamente la respuesta en memoria:

```
 550 DIM R$(1)
 555 INPUT "DESEAS OTRA PRUEBA?
 ";R$
 556 IF CODE R$=CODE "S" THEN RU
N
```

El programa se explica por sí mismo: si el primer carácter de la respuesta tiene un código (CODE) que es el mismo que el de S (es decir, es S), entonces el programa pasa nuevamente. Las entradas (INPUTS) nulas son rechazadas, entendiéndose que el usuario no desea jugar otra vez ya que, pulsando meramente NEWLINE proporciona una cadena vacía y el código (CODE) de las de este tipo es CERO como un espacio.

La comprobación de la primera letra de un INPUT (entrada) es bastante fácil como se acaba de ver. Más difícil resulta cuando se quiere comprobar todo el INPUT; es decir, ver si el usuario ha introducido algunos signos de puntuación o ha incluido letras en una entrada numérica. Veamos primero un INPUT alfabético. Los operadores de relación: <,>, > =, < =, y < >, son muy útiles en este caso. Consideremos un INPUT donde se requiere una palabra estrictamente y no debe entrarse ninguna otra cosa.

```
  10 INPUT A$
  15 IF A$="" THEN GO TO 10
  20 FOR A=1 TO LEN A$
  30 IF A$(A)<"A" OR A$(A)>"Z" T
HEN GO TO 10
  40 NEXT A
```

La línea 15 asegura que las entradas (INPUTS) nulas (como la simple pulsación del ENTER) son rechazadas. El bucle que empieza en la línea 20 explora todos los caracteres de la cadena del INPUT, de uno en uno, y si se encuentra alguno que no es una letra se indica que se vuelve a entrar nuevamente la cadena, ya que el programa salta hacia atrás a la línea 10. De esta manera, no se permiten espacios entre las palabras.

Cambie la línea 30 para que los espacios sean permitidos:

```
  30>IF (A$(A)<"A" OR A$(A)>"Z")
 AND A$(A)<>" " THEN GO TO 10
```

Puede fácilmente extenderse esta idea para que sean posibles los signos de puntuación, letras y espacios si se desea (números, claves, símbolos, etc., no se permiten) ampliando la idea en la línea 30. Resulta un poco más difícil detectar una determinada palabra en un INPUT. Así, por ejemplo, si se tiene una línea al final de un programa invitando al usuario a hacer otro paso de aquél y que en caso afirmativo, respondiendo "SI" vuelva a pasar el programa, es bastante sencillo poner el INPUT en un ciclo y analizar la palabra como sigue:

```
7010 INPUT "OTRA VEZ? ";A$
7020 FOR A=1 TO LEN A$-1
7030 IF A$(A TO A+1)="SI" THEN
RUN
7040 NEXT A
7050 STOP
```

Si se entra "SI" o "SI, POR FAVOR" el programa volverá a ejecutarse como se le solicita. Si la palabra entrada tiene una longitud inferior a la de la requerida (excepto la cadena ["string"] nula o vacía) se causará error debido a que la línea 7030 establece que el INPUT será, por lo menos, igual a la citada palabra. La cadena vacía hace que la longitud de A$ (LEN A$) sea igual a cero, por lo que la línea 7020 FOR A = 1TO 0, con lo que dicha cadena es ignorada y no surge el problema. Pruebe a entrar la palabra "SIETE", la rutina se vuelve a pasar porque ha detectado las dos letras "SI". Lo que se necesita es una rutina que detecte si el carácter a ambos lados de estas dos letras es algo que no sea una letra. Es preciso actuar con mucho cuidado para hacer esto porque no es posible analizar ambos lados si el fonema "SI" se presenta al principio o al final de la expresión de entrada y el intento de este examen produciría un mensaje de error. A continuación se presenta una rutina que se hace cargo de esto al añadir caracteres simulados al principio y al final de A$. Esta vez la palabra clave será "MAS".

```
7010 INPUT "OTRA VEZ? ";A$
7015 LET A$=" "+A$+" "
7020 FOR A=2 TO LEN A$-3
7030 IF A$(A TO A+2)="MAS" AND (
A$(A-1)<"A" OR A$(A-1)>"Z") AND
(A$(A+3)<"A" OR A$(A+3)="Z") THE
N RUN
7040 NEXT A
7050 STOP
```

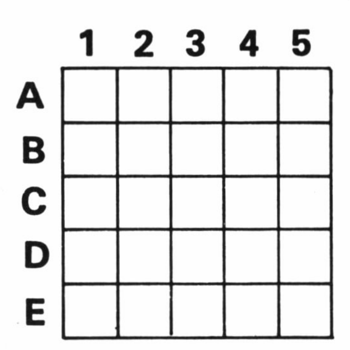
La rutina permite todas las longitudes del INPUT hasta el máximo posible. Si se desea cambiar la palabra clave en un programa vale

la pena asignarla a una variable o tener un INPUT en cualquier parte del programa para dicha palabra. Tendrán que hacerse las siguientes modificaciones a la rutina para utilizar una palabra clave diferente:

```
7010 INPUT "ENTRA PALABRA CLAVE"
;S$
7020 INPUT "ENTRA FRASE";A$
7040 LET A$=" "+A$+" "
7050 LET LS=LEN S$
7060 LET LA=LEN A$
7070 FOR A=2 TO LA-LS
7080 IF A$(A TO A+LS-1)=S$ AND (
A$(A-1)<"A" OR A$(A-1)>"Z") AND
(A$(A+LS)<"A" OR A$(A+LS)>"Z")
THEN RUN
7090 NEXT A
```

Si la rutina es demasiado larga y siempre se usa la misma palabra clave se puede evitar el empleo de S$ y L$ y poner dicha palabra con todas sus letras cuando haga falta, y sustituir todas las referencias a LS con la longitud de la repetida palabra. Véase el ejemplo anterior con "MAS".

Veamos ahora otro tipo de INPUT que se usa mucho en los juegos, tanto en los de tablero como en los de cuadrícula, y que utiliza coordenadas como las de los mapas. Por ejemplo, consideremos un tablero como el siguiente:



Las coordenadas se introducen en forma de una letra seguida por un número; por ejemplo, C3 si nos referimos a un cuadrado como en el juego de los barcos o C3B4 si se trata de ir de un cuadrado a otro como en el juego de las damas. Si se ha decidido que las coordenadas se van a introducir en la forma de una letra seguida por un número es muy probable que más pronto o más tarde, deliberada o acciden-

talmente, se apliquen en orden equivocado y se interrumpa el programa. Esta rutina detectará automáticamente si las dos cifras de una coordenada se han introducido invertidas y las ordena debidamente. Si se aplica al cuadrado representado anteriormente, para modificarlo con objeto de admitir otra escala de valores, cámbiense simplemente los caracteres entrecomillados en las líneas 30 y 40. La 50 se ha incluido exclusivamente para que pueda verse el efecto de la rutina, si hace al caso.

```
  10 INPUT A$
  20 IF LEN A$<2 THEN GO TO 10
  25 LET A$=A$( TO 2)
  30 IF A$(1)>="1" AND A$(1)<="5
" AND A$(2)>="A" AND A$(2)<="E"
THEN LET A$=A$(2)+A$(1)
  40 IF A$(1)<"A" OR A$(1)>"E" O
R A$(2)<"1" OR A$(2)>"5" THEN GO
 TO 10
  50 PRINT A$
```

La rutina es muy rápida de ejecutar y es muy difícil de romperla, aunque es seguro que algún lector inteligente encontrará la forma de hacerlo. Si usted encuentra la manera de vencer la rutina, modifíquela para evitar que el error vuelva a surgir.

Una rutina para coordenadas de cuatro cifras es algo más compleja. La idea de este INPUT es que se pueda entrar el número del cuadrado de donde se sale y el número del que va a ocuparse en una jugada. Por ejemplo, E3D4 significaría que se mueve una pieza desde el cuadrado E3 al D4. Permítasenos primeramente poner las letras y números en orden.

```
  10 INPUT A$
  20 IF LEN A$<4 THEN GO TO 10
  30 LET A$=A$( TO 4)
  40 IF A$(1)>="1" AND A$(1)<="5
" AND A$(2)>="A" AND A$(2)<="E"
THEN LET A$( TO 2)=A$(2)+A$(1)
  50 IF A$(3)="1" AND A$(3)<="5"
 AND A$(4)>="A" AND A$(4)<="E" T
HEN LET A$(3 TO )=A$(4)+A$(3)
  60 IF A$(1)<"A" OR A$(1)>"E" O
R A$(2)<"1" OR A$(2)>"5" OR A$(3
)<"A" OR A$(3)>"E" OR A$(4)<"1"
OR A$(4)>"5" THEN GO TO 10
  70 PRINT A$
```

Obsérvese que pueden acortarse ambas rutinas utilizando la instrucción de dimensiones DIM. Al primer programa es posible añadir:

5 DIM A$(2)

y omitir las líneas 20 y 25. Para el segundo programa agregue:

5 DIM A$(4)

y suprímanse las líneas 20 y 30. Lo que ambas versiones consiguen es asegurar que la cadena alfanumérica A$ no es ni más corta ni más larga de lo necesario. Si se entra un INPUT de más de cuatro caracteres en la segunda rutina, el resto es ignorado. Si tiene menos de cuatro se añaden espacios si se ha incorporado la línea 5 (rechazados en la línea 60), o rechazados en la línea 20 si se emplea la versión sin modificar. Habiendo clasificado las letras y números, vamos a ordenar los movimientos correctos y los incorrectos. Será preciso que se mire al tablero presentado un par de páginas atrás. Supongamos que tenemos una pieza del juego de damas en el cuadrado E3. Necesitamos determinar los movimientos correctos desde allí. Una pieza como la que tenemos solamente puede moverse hacia adelante en sentido diagonal. Los cuadrados en los que puede terminar son D2 ó D4. Antes de continuar la lectura, ¿es capaz de determinar la relación entre las coordenadas?

Puesto que la pieza sólo puede avanzar un cuadrado cada vez, tiene que terminar en otro cuya letra sea alfabéticamente la más próxima a E. Dado que en su ordenador los códigos (CODE) de las letras que se suceden unas a otras por dicho orden crecen o decrecen en una unidad, el código (CODE) de la letra D es uno menos que el de la E. Por consiguiente, si el código de la letra del cuadrado desde donde se inicia el movimiento no es una unidad mayor que el de la que corresponde al cuadrado de llegada, el movimiento es incorrecto. El número del cuadrado de partida debe ser una unidad mayor o menor que el de llegada, por lo que nos encontramos con el siguiente planteamiento:

```
   65 IF (CODE A$(1)<>CODE A$(3)+
1) OR (CODE A$(2)<>CODE A$(4)+1)
 OR (CODE A$(2)= CODE A$(4)-1)
THEN GO TO 10
```

Resulta obvio que será necesario adaptar estas rutinas a los correspondientes programas y que sólo se pretende mostrar su propio fundamento para que sobre él se puedan elaborar las rutinas que sean necesarias. También sirven para demostrar el enfoque preciso que ayuda a resolver los problemas de esta clase. Sugerimos las siguientes recomendaciones por considerarlas de valor:

(I) Determine exactamente lo que quiere realizar.

(II) Fije con exactitud lo que está permitido y algo de lo que no lo está (las cadenas [''string''] vacías, por ejemplo).

(III) ¿Cómo pueden impedirse las acciones prohibidas, o rechazarlas cuando se incurre en ellas?

(IV) Compruebe mentalmente si su rutina realiza lo pretendido por medio de un par de ejemplos.

(V) Si está satisfecho con su rutina, introdúzcala al ordenador y pruébela con algunos valores o caracteres permitidos para saber si existe algo que impida su entrada.
Tras el paso anterior con éxito compruébese la rutina con toda clase de entradas (INPUTS). (Por ejemplo, trate de entrar una coordenada inexistente, como F9 en las rutinas anteriores.) Ahora ya se está preparado para la prueba más importante.

(VI) Permita que un amigo someta a prueba la rutina con órdenes que la desconcierten. Las rutinas anteriores tienen un error pero no vamos a decir cuál es. Se lo dejamos a usted como ejercicio.

Veamos finalmente las entradas (INPUT) numéricas. Deje en blanco su ordenador con la instrucción NEW y entre lo siguiente:

```
10 INPUT A
20 GO TO 10
```

Ejecute (RUN) este programita y vea si lo puede romper de alguna manera; no debe ser demasiado difícil. Pruebe a entrar una letra, o un STOP, o un número demasiado grande o demasiado pequeño para la capacidad del ordenador: o bien, una palabra clave o un signo aritmético tal como '' + ''.

Los signos aritméticos hacen que el ordenador presente la indicación de un error de sintaxis aunque no detienen el programa. Claves y símbolos también producen esto, si bien las letras hacen que el programa se detenga con un código 2 de error, indicando que se ha utilizado una variable no definida. Se ha dicho que las variables (sin adjetivar) son siempre numéricas, ¿por qué al error derivado de la entrada de una letra lo atribuimos a una variable sin definir? No es una incorrección. Cuando se entra una letra en respuesta a un INPUT numérico el ordenador piensa que se está introduciendo una variable y esto puede ser muy útil en ocasiones. Con el mismo programa entre un 1 como primer INPUT, la segunda vez introduzca A y... ¡es aceptada! Lo que el ordenador ha hecho es buscar el valor de A y asignárselo a la variable de este nombre. En otras palabras, no ha cambiado el valor de A. Ahora entre STOP. El programa se detiene con un men-

saje de error. Pruebe a introducir PRINT A, y aparecerá el 1 de forma que el programa se ha detenido *antes* de actualizar el valor de A. En realidad, cuando se interrumpe un INPUT numérico, el ordenador, por lo general, retiene el valor previo de la variable. No es que esto sea muy útil pero, en determinadas circunstancias, si no se reinicia el programa, la variable tiene un valor.

La mejor forma de hacer frente a estos problemas es mediante el uso de variables alfanuméricas ("string") como INPUTS, evaluándolas con VAL. Pruebe:

```
10 INPUT A$
20 LET A=VAL A$
30 PRINT A
40 GO TO 10
```

Encontrará bastante fácil interrumpir el programa. Muchas cosas válidas de las entradas (INPUTS) numéricas parecen suceder con esta pequeña rutina. Sin embargo, la ventaja de este método es que no se interrumpe hasta que se aplica el enunciado VAL si existe un error. Puede procesarse la cadena ("string") antes de la aplicación de VAL después de la entrada del INPUT y eliminarse los errores antes de que se produzca la interrupción, es decir, se puede comprobar la cadena. Lo importante a recordar es que VAL puede aplicarse con cualquier cosa numérica no solamente con números. Pruebe lo que sigue:

PRINT VAL "RND"
PRINT VAL "SGN −7"
PRINT VAL "A*2" (esto sólo es posible si se ha definido previamente A)
PRINT VAL "COS1"

Los nombres de variables no definidas son el azote de VAL, junto con las instrucciones no numéricas, las claves o los símbolos, que han de eliminarse antes de la aplicación de dicha instrucción. El caso más sencillo es en el que sólo se permiten las entradas (INPUT) numéricas y puede hacerse como sigue:

```
  10 INPUT A$
  20 FOR F=1 TO LEN A$
  30 IF A$(F)>"9" OR A$(F)<"0" T
HEN GO TO 10
  40 NEXT F
  50 LET A=VAL A$
  60 PRINT A
```

¿Puede verse de inmediato lo que provocaría esta rutina? Nuestra vieja amiga, la cadena nula o vacía, tendría como bucle FOR F = 1 TO 0 por lo que se ignoraría y resultaría inútil. La solución debería ser añadir: 15 IF A$ = "" THEN GO TO 10. La rutina hace que vuelva a entrar el número si lo que se ha aplicado no lo era. Es posible extender la idea para permitir signos aritméticos y nombres de variables si se desea pero tiene tan poco uso que no vale la pena.

```
 30 IF (A$(F)>"9" OR A$(F)<"0")
AND (A$(F)<>"+" AND A$(F)<>" ")
THEN GO TO 10
```

Esta rutina permite entrar los signos de adición y potenciación. Para admitir funciones adicionales agréguense simplemente al segundo espacio entre paréntesis que se halla enlazado con el anterior por AND. Esto no es que sea tremendamente útil pero algún día puede encontrarlo aplicable. Volveremos a la instrucción VAL y a otras funciones de análisis de cadenas más adelante pero ahora necesitamos examinar instrucciones fundamentales de la capacidad de "pensar" del ordenador.

GO TO

Una importante facultad de la programación es la de poder enlazar diferentes partes de un programa durante su ejecución. Sin esto, el programa siempre se ejecutará según el orden numérico de sus líneas y al llegar a la de número más alto se detendría. Una instrucción que permite movimientos a voluntad dentro del programa es el GO TO. Ir a la instrucción GO TO se compone de un número de orden de línea seguido por GO TO y otro número de línea o un cálculo (como GO TO 2*X, o GO TO 200 + 340).

Si el ordenador se encontrara 140 GO TO 190, saltaría inmediatamente desde la línea 140 a la 190. Esto se llama un salto incondicional. Es decir, es un salto que no depende de la existencia de una condición. Una vez en la línea 190, el programa continúa su ejecución por orden hasta el final o se encuentra con otra línea que lo envía a cualquier otra parte.

Se puede utilizar el GO TO para producir programas que no terminan. Son muy efectivos, especialmente al final de un juego. Ejercítese el siguiente para verlo en acción:

```
   10 PRINT INK RND*6;"..HAS GANA
DO..";
   20 POKE 23692,-1
   30 BEEP .01,RND*50
   40 GO TO 10
```

La instrucción IF... THEN GO TO

La instrucción IF realiza una función similar a GO TO pero solamente desviará el programa si se cumplen ciertas condiciones. Ello origina un salto condicional. La instrucción IF/THEN se compone de un número de línea seguido de las palabras IF/THEN/GO TO (SI/ENTONCES... IR A) separadas por una relación que debe determinarse antes de dejar la línea. Existen seis *operadores de relación* que pueden utilizarse para comparar dos variables. Son éstos:

= igual a
> mayor que
< menor que
<> distinto que
> = mayor o igual a
< = menor o igual a

Estos operadores se utilizan con las instrucciones IF... THEN (SI [condición] ENTONCES) para formar la condición a determinar.

He aquí un ejemplo: 70 IF Z > = 10 THEN GO TO 100.

Esto será interpretado por el ordenador en el sentido de que SI el valor de la variable Z es mayor o igual que 10, el programa se desviará a la instrucción de la línea 100. Si Z es menor que 10, el programa continuará su ejecución normal pasando a la línea siguiente, la 80. Con ello se proporciona al ordenador la capacidad de tomar decisiones, el

verdadero origen de la aparente facultad de pensar de la máquina.

Como probablemente usted ya habrá descubierto, el ordenador no es indeciso (a menos que se le diga que lo sea). En cada caso toma una firme decisión de hacer o de no hacer algo. Lo que realmente hace depende de lo que se le diga, generalmente después de la palabra THEN de la línea. Vamos a ilustrarlo con un programa sencillo para presentar en pantalla el número que se acaba de entrar en forma literal en lugar de hacerlo con los signos numéricos.

```
   10 INPUT INK RND*6;"ENTRA UN
NUMERO DEL 1 AL 3 ";A
   30 IF A=1 THEN PRINT "UNO"
   40 IF A=2 THEN PRINT "DOS"
   50 IF A=3 THEN PRINT "TRES"
   60 GO TO 10
```

No necesariamente ha de estarse limitado por una condición entre el IF (SI) y el THEN (ENTONCES). Para seguir con el ejemplo anterior supongamos que a usted se le permitiera ir a casa a las cinco tan sólo si hubiera terminado su trabajo:

SI (IF) son las cinco de la tarde Y (AND) ha terminado su trabajo ENTONCES (THEN) váyase a casa. Cuando se desea reunir dos o más expresiones de condición con otra como "usted ha acabado su trabajo", pueden utilizarse tres palabras de enlace para lograrlo. Estas son: Y (AND), O (OR) y NO (NOT). Si se tiene una expresión condicional con la conjunción Y (AND) uniendo ambas partes, entonces el ordenador hace algo si ambas partes son verdaderas. Si son las cinco de la tarde pero usted no ha terminado su trabajo, entonces no se puede ir a casa, por ejemplo.

Para ilustrar el análisis de VERDADERO (TRUE) y FALSO (FALSE), probemos este programa:

```
   10 INPUT A
   20 INPUT B
   30 IF A=1 AND B=1 THEN PRINT "
VERDADERO"
   40 GO TO 10
```

Trate de entrar diferentes valores y vea los resultados. Intente cambiar los valores de la línea 30 y observe el efecto que produce. Tome nota de sus respuestas hasta que comprenda lo que pasa.

Veamos lo que sucede con el OR (O). Pensemos en la conjunción disyuntiva en relación con la expresión "si (IF) son las cinco de la tarde o (OR) el jefe le autoriza, entonces (THEN) puede irse a casa". Esto es hacer algo cuando una de las alternativas es cierta. Mejor aún,

hacer algo cuando, por lo menos, una de las alternativas es cierta ya que carece de importancia cuántas pudieran serlo (incluso sería posible que lo fueran todas). Así que usted se va a su casa a las cinco de cualquier manera, cualquiera que sea la circunstancia que le permita marcharse. Pruebe a experimentar con el siguiente programa tal como lo hizo con el anterior.

```
  10 INPUT A
  20 INPUT B
  30 IF A=1 OR B=1 THEN PRINT "
VERDADERO"
  40 GO TO 10
```

La última de estas palabras de condición es NO (NOT). No une expresiones como las anteriores pero cambia su significado. Estudie esto:

SI (IF) el director NO (NOT) ha dicho que se vaya a casa, ENTONCES (THEN) quédese trabajando.

Significa que, a menos que se le haya dicho que se vaya, ha de permanecer en el trabajo. Lo que pasa es que el ordenador mira la expresión y decide que si no es verdad, entonces hace algo (con esta finalidad ignora el NO para decidir lo que es verdad y lo que no lo es). Es decir, SI NO... ENTONCES (IF NOT... THEN) es verdad cuando lo que siga al NO sea falso. Se hace algo sólo cuando una condición no se cumple.

Pruebe esto:

```
  10 INPUT A
  20 INPUT B
  30 IF A<>B THEN PRINT "VERDADE
RO"
  40 GO TO 10
```

Esto puede parecer confuso al principio pero si se experimenta con los valores de A y B se observará un conjunto de resultados que ilustran el funcionamiento del NO (NOT).

Se habrá advertido que hemos utilizado el signo igual (=) en todos los ejemplos hasta ahora. Recuérdese que éste es sólo uno de los seis *operadores de relación.* He aquí nuevamente la lista de los seis que utiliza su ordenador:

< menor que
> mayor que

< = menor o igual a
> = mayor o igual a
<> distinto a
= igual a

Cambie los programas de forma que se utilicen todos los OPERADORES DE RELACION. Juegue con estos programas hasta que sea capaz de predecir lo que va a suceder cada vez. Pruebe con combinaciones de Y, O y NO (AND, OR, NOT) y vea en qué orden son manejados. Vea si puede determinar cómo cambiar el resultado encerrando las expresiones entre paréntesis. Observe que no siempre se puede, de manera que, si ciertas expresiones le dan problemas, déjelas y pruebe con otras. El orden de la evaluación viene determinado por las *prioridades*, de las que nos ocuparemos con detalle más adelante.

Se pueden aplicar las expresiones condicionales a las cadenas alfanuméricas así como a los números.

```
   10 INPUT A$
   20 IF A$<>"PACO PICAPIEDRA"
THEN NEW
```

Ejecute el programa y vea lo que sucede. La primera vez que lo haga entre el nombre PACO PICAPIEDRA en mayúsculas. El programa se detendrá normalmente. Poco interesante. Vuelva a ejecutarlo otra vez y en esta ocasión entre su propio nombre (si resulta que verdaderamente se llama PACO PICAPIEDRA, ponga el nombre de otro). Ahora el programa se autodestruirá debido al NEW de la línea 20. Si sustituye su nombre o un número en clave por PACO PICAPIEDRA, tendrá un programa que sólo trabajará para usted o para los que conocen la clave y se autodestruirá si alguien intenta utilizarlo.

Veamos ahora los valores en las expresiones condicionales. Primeramente utilizaremos los operadores relacionales. Observará que verdadero se representa por 1 y falso por 0.

```
10 INPUT A
20 INPUT B
30 LET X=(A=B)
40 PRINT X
50 GO TO 10
```

No son verdaderamente necesarios los paréntesis en la línea 30 pero ayudan a clarificar el significado. La expresión entre paréntesis se reduce a 0 o 1 que dependen de los valores que se introduzcan. Pruebe a utilizar los seis operadores relacionales y tome nota de los resultados.

Obtendrá 0 ó 1 cada vez, lo que le hará pensar que esto es un poco restrictivo. En realidad, como este valor de 0 ó 1 puede considerarse como un número, es posible manipularlos como se hace con ellos. La mejor manera de manipular números es multiplicarlos, ya que esto cambiará los valores verdaderos pero no los falsos (lo que se multiplica por cero es 0). Cambie el programa para formar el siguiente:

```
10 INPUT A
20 INPUT B
30 LET X=(A=B)*2
40 PRINT X
50 GO TO 10
```

Esta vez obtendrá el valor 0 para las expresiones falsas y un 2 para las verdaderas. El punto interesante es que estos valores de expresiones condicionales son números y pueden tratarse como tales lo que resulta muy útil. A continuación, sigue el programa de un sencillo juego, CAZADOR DE BURBUJAS, para ilustrar la utilización de lo que hemos estado tratando.

```
  10 REM *CAZADOR DE BURBUJAS*
  20 LET S=0
  30 FOR J=1 TO 9
  40 PRINT AT 10,3*J; INK J/2;J
  50 NEXT J
  60 FOR G=30 TO 1 STEP -1
  70 PRINT AT 5,3; INK 6; PAPER
2; FLASH 1;" TIEMPO>";G;"     TAN
TEO>";S;" "
  80 LET A=INT (RND*9)+1
  90 PRINT AT 9,3*A; FLASH 1; IN
K 2;"█"
 100 FOR H=1 TO 24
 110 BEEP .01,50/H
 120 LET A$=INKEY$
 130 IF A$<>"" THEN GO TO 170
 140 NEXT H
 150 LET G=G-1
 170 LET S=S+(A$=STR$ A)*A
 180 PRINT AT 9,3*A;" "
 190 NEXT G
 200 PRINT AT 5,3; INK 6; PAPER
2; FLASH 1;" TIEMPO>";0;"    TANT
EO>"; PAPER 6; INK 2;S;" "
 210 PRINT AT 13,3; FLASH 1; INK
 0; PAPER 6; BRIGHT 1;"SE ACABO,
 AMIGOS!!"
```

La idea del juego consiste en pulsar la tecla del número bajo la burbuja móvil. Por ejemplo, si se para sobre el 3 ha de pulsarse el número 3 y se adjudica como puntuación el valor del número. En este

caso se añade 3 al tanteo. Los intentos que quedan se presentan continuamente en la pantalla igual que el tanteo. La línea 170 es la que nos interesa de momento. Aquí si el enunciado STR$ A (valor de A convertido en una cadena alfanumérica [''string''] de forma que pueda compararse con la tecla pulsada) es igual a la tecla accionada, el valor lógico se hace 1 porque la expresión es verdadera. Cualquiera que sea el valor, se multiplica por A. Si es 0, el tanteo no cambia: si es 1, el tanteo se incrementa en 1*A. El tanteo es contado por la variable S. El número de ensayos que quedan lo determina G.

Pasemos ahora a examinar los valores de las expresiones condicionales en las que intervienen las operaciones lógicas Y, O y NO (AND, OR, NOT). El valor que toma la operación lógica X e Y (X AND Y) es el siguiente:

X si Y es verdadera (no cero)
0 si Y es falsa (cero)

Las variables X e Y pueden ser expresiones como X = 2 ó Y = 2*B. Una aplicación común es para el control del movimiento en la pantalla. Muchos juegos utilizan las teclas de flechas cursoras para dicho movimiento, consiguiendo así el desplazamiento hacia la izquierda o la derecha de algún elemento de presentación:

```
  10 LET X=15
  20 IF INKEY$="5" AND X>1 THEN
LET X=X-2
  30 IF INKEY$="8" AND X<30 THEN
 LET X=X+2
  40 PRINT AT 21,X; INK 2;"█";AT
 21,X;" "
  50 GO TO 20
```

Con este programa se mueve un cuadradito rojo en saltos de dos columnas a lo largo de la hilera del fondo de la pantalla. Puede lograrse lo mismo con:

```
  10 LET X=15
  20 LET X=X-(INKEY$="5" AND X>1
) *2+(INKEY$="8" AND X<30) *2
  40 PRINT AT 21,X; INK 2;"█";AT
 21,X;" "
  50 GO TO 20
```

O con:

```
   10 LET X=15
   20 LET X=X-(2 AND INKEY$="5" A
ND X>1)+(2 AND INKEY$="8" AND X<
30)
   40 PRINT AT 21,X; INK 2;"█";AT
 21,X;" "
   50 GO TO 20
```

El punto a señalar en los dos últimos programas es que las expresiones entre paréntesis toman el valor del número delante del primer AND (Y) si todas las expresiones que están después de él son verdaderas. Compárense con las operaciones lógicas X e Y de las que acabamos de tratar. Aquí X es un número (2 en este caso) y no una expresión. Puede pensarse de la línea 20 del programa anterior como sigue: 20 LET X = X — (2 si se pulsa la tecla "5" y si el valor de X es mayor que 1; en otro caso, 0) + (2 si se pulsa la tecla "8" y el valor de X es menor que 30; en otro caso, 0).

Pudiera preguntarse para qué tantas complicaciones para hacer algo que pudiera lograrse exactamente igual con una serie de líneas IF... THEN. La respuesta es que si aquellas expresiones se usan adecuadamente y en las circunstancias oportunas, se pueden sustituir varias líneas del programa con una expresión condicional larga, ahorrando capacidad de memoria y posiblemente haciendo que dicho programa se ejecute más rápidamente. Además, al familiarizarse con estas expresiones condicionales es posible encontrar que algunas veces pueden clarificar realmente listados sobre un largo juego de instrucciones IF... THEN.

Pasemos ahora a la operación O (OR).

X o Y toma el valor 1 si Y es no 0 (verdadero)
y el valor X si Y es cero (falso)

Supongamos que el cobrador de un autobús desease un programa para saber el precio que ha de cobrar a un escolar y que la edad límite para la tarifa reducida es de 14 años.

```
   10 INPUT "ENTRAR TARIFA ";TARI
FA
   20 INPUT "ENTRAR EDAD ";EDAD
   30 LET TARIFA=TARIFA*(0.5 OR
EDAD>14)
   40 PRINT "EL PRECIO ES ";TARI
FA
```

Las líneas 10 y 20 piden que se introduzca la tarifa normal de adul-

tos y la edad del pasajero. Para entender esto un poco mejor convirtámoslo en un lenguaje más claro:

SEA TARIFA = TARIFA*(0,5 a menos que la edad sea superior a 14)

Si la expresión que sigue a O dentro del paréntesis es verdadera, toda la expresión del paréntesis toma el valor 1. Sin embargo, si lo que sigue a O es falso (edad menor que 14 años), la expresión encerrada toma el valor que hay delante de O. Este número (0,5) puede también ser una variable si se desea. En conjunto, esta rutina no tiene mucho que ofrecer en relación con la siguiente:

30 IF EDAD < = 14 THEN LET TARIFA = TARIFA*0,5
(Si edad < = 14, entonces sea tarifa = tarifa por 0,5)

Sin embargo, si se dispone de varias tarifas el método del O (OR) puede extenderse para evaluar todas las categorías en una línea.

"NO X" (NOT X) toma el valor de 0 si la relación X es verdadera, y el valor 1 si es falsa. La mejor forma de ilustrar esto es con el ejemplo que sigue:

```
   10 INPUT A
   20 INPUT B
   30 PRINT A;TAB 4;B;TAB 8;NOT
A=B
   40 GO TO 10
```

Lo que se verá en la pantalla son los números que se introdujeron en las líneas 10 y 20, seguidos de un 0 ó un 1. De los resultados obtenidos, véase si se puede determinar qué relación entre A y B produce los respectivos valores de la columna tercera. Pruébense los otros operadores de relación en lugar del signo < de la línea 30.

Finalmente, echemos una ojeada a dos interesantes pequeñas rarezas. Considérese primero esta línea:

```
   10 IF A=1 THEN IF B=2 THEN
PRINT "VERDADERO"
```

Es enteramente lo mismo que:

```
   10 IF A=1 AND B=2 THEN PRINT
"VERDADERO"
```

excepto que ésta requiere memoria extra. Existe una pequeña diferencia en que si no se ha definido previamente B, la versión que usa AND se detendrá con mensaje 2. Sin embargo, si la primera parte de la otra

versión es falsa, entonces el programa desatiende el resto de la línea. Es posible que se encuentre una aplicación de esto.

La segunda rareza no lo es tal sino algo que desconocen muchas personas. Pruebe estos programas:

```
10 INPUT A
20 IF A THEN PRINT A
```

```
10 INPUT A
20 IF NOT A THEN PRINT A
```

Es posible que no se espere que funcionen estos programas ya que no hay operadores de relación para comparar A con algo. Aquí, sin embargo, el valor de A se considera verdadero si no es 0, o 0 si se utiliza el NOT. Igual que con todo lo demás de esta sección, experiméntese con los ejemplos hasta que se entienda perfectamente lo que hace cada rutina. Se observará que estas instrucciones pueden consituir poderosas ayudar de programación, y se verá incrementada considerablemente la eficacia en esta actividad.

Se puede utilizar IF/THEN GO TO para terminar un mensaje de "condición de ganador" después de un cierto número de ciclos. Entre y ejecute lo siguiente:

```
10 LET X=0
20 PRINT "HAS GANADO ";
30 LET X=X+1
40 IF X<25 THEN GO TO 20
```

Esto asegura que se presente el mensaje "usted ha ganado" un limitado número de veces.

Como se ha visto, IF/THEN no sólo se utiliza para enlazar con otras líneas. Elimine el programa con el enunciado NEW y aplique el siguiente. Verá que tiene un efecto similar, aunque el IF... no envía el programa a un número de línea.

```
  10 LET X=0
  20 LET X=X+1
  30 IF X<25 THEN PRINT  "HAS GA
NADO ";
  40 GO TO 20
```

Este programa no es tan útil como el otro ya que no termina aunque haya dejado de presentar el "usted ha ganado". Puede fácilmente descubrirlo ejecutándolo, pulsando después BREAK (detención) y a continuación PRINT X, ENTER.

Vale la pena quizá mencionar que el ordenador es un ente bastante dogmático. Si se especifica que una rama de la programación ha de ejecutarse sólo si el valor de Z, por ejemplo, es igual a 6, el programa continuará en un bucle sin fin si Z no es exactamente igual a 6, no importa lo mucho que se le aproxime (como con 5,999999). Si se piensa que el valor pudiera ser fraccionalmente diferente con respecto al que se establece para la desviación, asegúrese que se especifica que el operador de relación debe ser mayor que 5,5, por ejemplo, o igual o mayor que 5,9, en lugar de hacerlo justamente igual a 6.

IF/THEN/ELSE

Muchos dialectos del BASIC incluyen la opción ELSE (SI/ENTONCES/EN OTRO CASO) utilizada en la instrucción IF... THEN... ELSE. No existe esta función en el BASIC de nuestro Spectrum pero es posible lograr su lógica. Es una variación muy útil del condicionante IF. El ordenador puede programarse para hacer algo si la condición sometida a prueba es verdadera y realizar otra cosa (ELSE) además que no sea pasar meramente a la línea siguiente, si la mencionada condición es falsa.

Se puede utilizar la siguiente sustitución del IF... THEN... ELSE para producir algunos gráficos muy interesantes. Basta simplemente con introducir en la línea 60 la función que se desea representar. No es éste el método más eficaz de programación pero es útil como medio de demostrar la sustitución de la instrucción que consideramos. Al ejecutar el programa, se calcula el valor de K cada vez que se alcanza la línea 60. La 70 busca el valor de K y presenta en la pantalla un 0 si K es igual o mayor que 0,5 y un paro total si es menor que este valor. Esto es lo mismo que si se dijese Si (IF) K es igual o mayor que 0,5 presente un "0"; en otro caso, un ".". Cada uno de los distintos gráficos utiliza diferentes valores de K, generados en la línea 60. La condición analizada en la línea 70 también varía. Ejecute el ejemplo que sigue utilizando el símbolo del gráfico que se desee y después genere unos pocos de su invención. Es probable que haya que cambiar la escala para ciertas funciones.

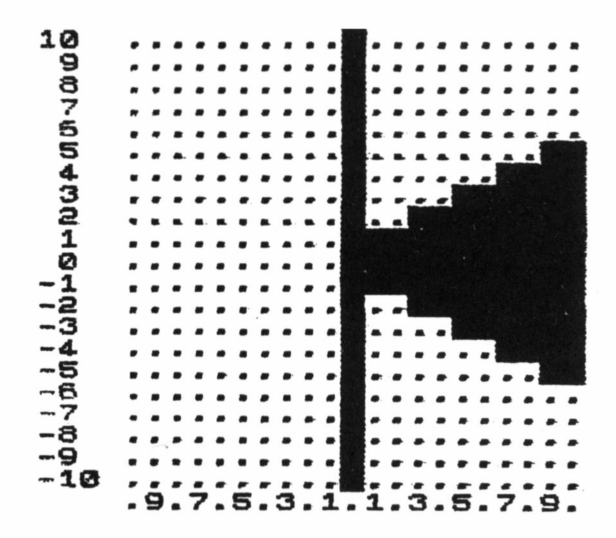
```
  10 REM GENERADOR DE GRAFICOS
  20 FOR Y=10 TO -10 STEP -1
  30 IF Y<>10 AND Y<>-10 AND Y>-
1 THEN PRINT " ";
```

```
  40 PRINT Y;TAB 4;
  50 FOR X=-10 TO 10
  60 LET K=Y-X*X/2+7
  70 PRINT ("0" AND K>=.5)+("."
AND K<.5);
  80 NEXT X
  90 PRINT
 100 NEXT Y
 110 PRINT TAB 4;".9.7.5.3.1.1.3
.5.7.9."
```

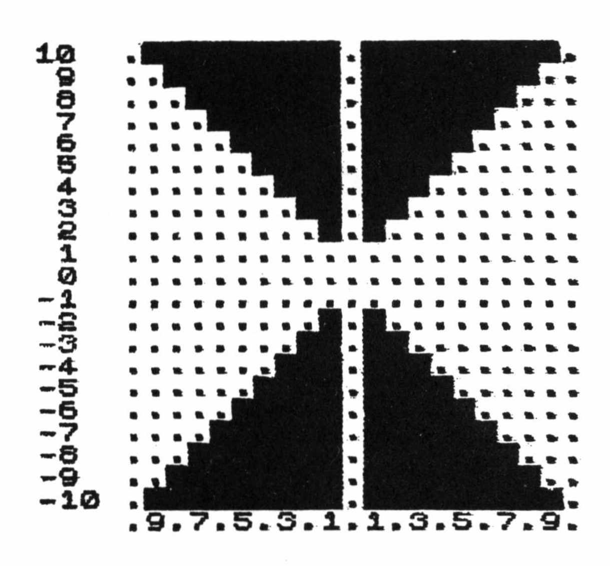
```
 10   .....00000000000.....
  9   .....00000000000.....
  8   .....00000000000.....
  7   .....00000000000.....
  6   .....00000000000.....
  5   ......000000000......
  4   ......000000000......
  3   ......000000000......
  2   ......000000000......
  1   .......0000000.......
  0   .......0000000.......
 -1   .......0000000.......
 -2   .......0000000.......
 -3   ........00000........
 -4   ........00000........
 -5   .........000.........
 -6   .........000.........
 -7   .....................
 -8   .....................
 -9   .....................
-10   .....................
      .9.7.5.3.1.1.3.5.7.9.
```

```
  9   .....00000000000.....
  8   ......000000000......
  7   ......000000000......
  6   ......000000000......
  5   .......0000000.......
  4   .......0000000.......
  3   ........00000........
  2   ........00000........
  1   .........000.........
  0   .....................
 -1   .........000.........
 -2   ........00000........
 -3   ........00000........
 -4   .......0000000.......
 -5   .......0000000.......
 -6   ......000000000......
 -7   ......000000000......
 -8   ......000000000......
 -9   .....00000000000.....
-10   .....00000000000.....
      .9.7.5.3.1.1.3.5.7.9.
```

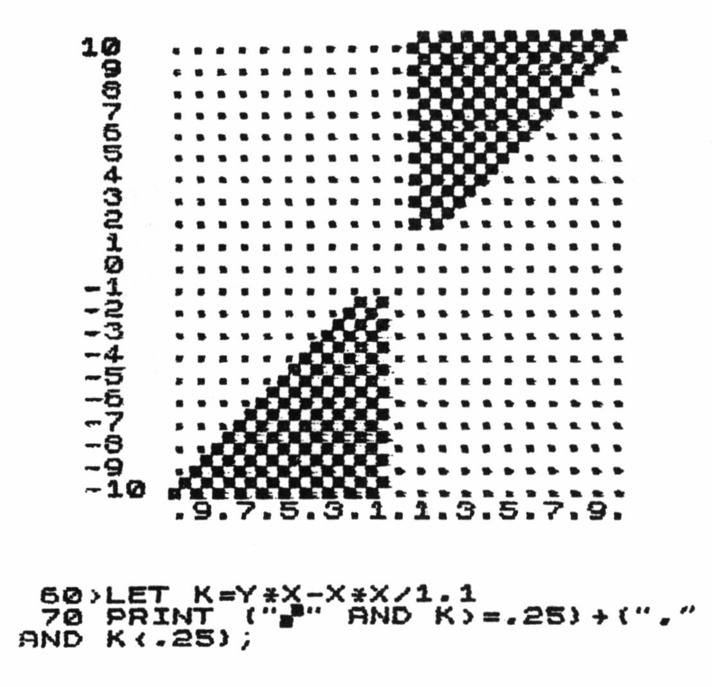
```
60>LET K=3*ABS (Y)-X*X
```




```
   60>LET K=SQR (ABS (Y*X*2)) -X
   70 PRINT ("█" AND K<=.5)+("."
AND K>.5);
```

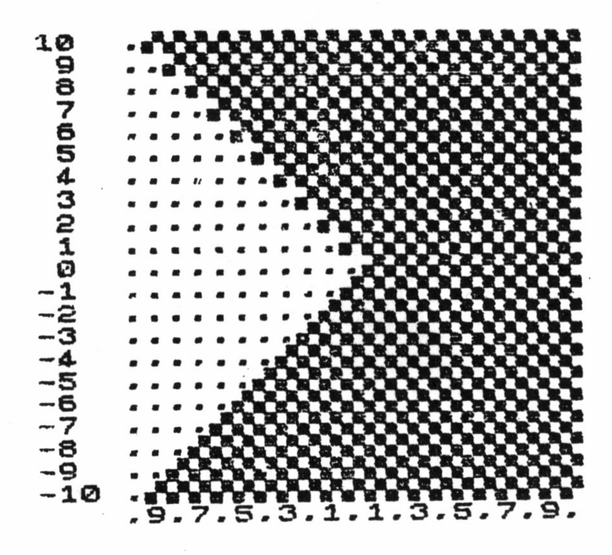



```
   60>LET K=ABS (Y*X) -X*X
   70 PRINT ("█" AND K>=.5)+("."
AND K<.5);
```




```
60>LET K=Y*X-X*X/1.1
 70 PRINT ("■" AND K>=.25)+("."
AND K<.25);
```

Trate de determinar qué habría que poner en la línea 60 para producir el gráfico que sigue:



Bucles FOR/NEXT

Estos bucles son elementos útiles adicionales del lenguaje BASIC. Merece la pena que se estudien ahora ya que las últimas series de programas dependían en gran manera en tales bucles. El de Y que empezaba en la línea 20 y terminaba en la 100; y el de X, desde la 50 a la 80. Debido a que son ligeramente más complejos que los bucles de este tipo más sencillos, no trataremos de ellos de momento.

Un bucle FOR/NEXT ("Para..., el siguiente") se compone de dos manifestaciones utilizadas para controlar una serie de ciclos en una parte del programa. El bucle empieza con FOR y especifica cuántas veces ha de ejecutarse. NEXT aparece al final de la secuencia volviendo el programa a la línea que sigue a la que contiene la instrucción FOR.

La instrucción FOR se compone del número de línea, a continuación el FOR, una variable numérica (una sola letra), el signo igual, una expresión numérica (un número o una variable numérica previamente asignada), la palabra TO (A) y, finalmente, otra expresión numérica (el número de la variable numérica asignada anteriormente) que es diferente de la primera. Puede parecer esto increíblemente complicado. Sin embargo, esto es muy sencillo.

La línea FOR dice:

100 FOR J = 1 TO 100

La línea NEXT, que termina el bucle, tiene la forma:

200 NEXT J

La instrucción NEXT se compone de un número de línea, la palabra NEXT ("siguiente valor de...") y la variable de control que se estableció en la instrucción FOR anteriormente. La instrucción NEXT se utiliza solamente para decir al ordenador cuándo ha de detenerse la secuencia del programa que ha estado repitiéndose. Cuando el valor de la variable de control (J) alcanza el establecido en la instrucción FOR (la segunda variable numérica) el programa pasa por el bucle por última vez y continúa en la línea que sigue a la que contiene la palabra NEXT.

Aplique y ejecute este programa:

```
10 FOR A=1 TO 10
20 PRINT TAB 4;A;TAB 8;A*A
30 NEXT A
```

```
1   1
2   4
3   9
4   16
5   25
6   36
7   49
8   64
9   81
10  100
```

La variable de control es A, y la línea 20 presenta en pantalla A y A^2. Obsérvese que los límites del bucle de control están explícitos en la línea 10 (1 TO 10).

Préstese atención al ejemplo siguiente:

```
10 LET A=5
20 LET B=16
30 FOR C=A TO B
40 PRINT TAB 4;C;TAB 8;C/10;TA
B 14;C/A
50 NEXT C
```

```
5   0.5  1
6   0.6  1.2
7   0.7  1.4
8   0.8  1.6
9   0.9  1.8
10  1    2
11  1.1  2.2
12  1.2  2.4
13  1.3  2.6
14  1.4  2.8
15  1.5  3
16  1.6  3.2
```

Se habrá advertido que en este programa los límites del bucle FOR/NEXT son dos variables, A y B, que han sido previamente definidas. Encontrará que hay muchos programas donde se deseará un bucle FOR/NEXT limitado por los resultados de cosas que han ocurrido en otra parte del mismo.

Bucles anidados

Como acabamos de ver, los bucles FOR/NEXT nos permiten alterar el valor de una variable (en saltos de una unidad en los casos que hemos estudiado) para repetir una serie programada de hechos, un cierto número de veces. Supongamos ahora que hubiera dos o más variables con las que operar. Entonces sería preciso cambiar el valor de todas ellas. Esto puede hacerse fácilmente por medio de los bucles anidados, en los que uno de ellos, controlado por una de las instrucciones FOR/NEXT, opera dentro del otro.

Ejecútese el siguiente programa que inserta el ciclo B en el A.

```
   10 REM BUCLES ANIDADOS
   30 FOR A=1 TO 12
   40 FOR B=1 TO 12
   50 PRINT TAB 8;B;" POR ";A;" E
S ";A*B
   60 NEXT B
   70 PRINT
   80 POKE 23692,-1
   90 NEXT A
```

Al ejecutarlo verá que ha presentado la tabla de multiplicar desde 1 × 1 hasta 12 × 12. Parte de la presentación es:

```
8 POR 4 IGUAL A 32
9 POR 4 IGUAL A 36
10 POR 4 IGUAL A 40
11 POR 4 IGUAL A 44
12 POR 4 IGUAL A 48

1 POR 5 IGUAL A 5
2 POR 5 IGUAL A 10
3 POR 5 IGUAL A 15
4 POR 5 IGUAL A 20
5 POR 5 IGUAL A 25
6 POR 5 IGUAL A 30
7 POR 5 IGUAL A 35
8 POR 5 IGUAL A 40
9 POR 5 IGUAL A 45
```

En este programa, la variable de control A permanece en uno mientras que el bucle controlado por B va de uno a doce. Después de la instrucción PRINT (línea 70), la variable de control A aumenta en una unidad, y se repite nuevamente el bucle B, con A igual a 2 en

esta ocasión; y así sucesivamente, hasta que el bucle B se ha completado con A igual a 12. No existe ninguna razón por la que hayan de anidarse solamente dos bucles.

Es de importancia esencial el que las variables de control de los bucles anidados se hallen en orden correcto; es decir, el primer bucle empezado es el último en terminar. Trátese de intercambiar las líneas 60 y 90 de este programa y vea lo que sucede.

Esto es parte del resultado, obviamente no lo que se pretendía.

```
1 POR 12 IGUAL A 12
2 POR 13 IGUAL A 26
3 POR 14 IGUAL A 42
4 POR 15 IGUAL A 60
5 POR 16 IGUAL A 80
6 POR 17 IGUAL A 102
7 POR 18 IGUAL A 126
8 POR 19 IGUAL A 152
9 POR 20 IGUAL A 180
10 POR 21 IGUAL A 210
11 POR 22 IGUAL A 242
12 POR 23 IGUAL A 276
```

Utilícese la misma variable para cuantos fines quiera, especialmente cuando se empleen bucles FOR/NEXT. No se emplee otra letra como nombre para un segundo bucle si ya se ha terminado con el previo, pues con ello se derrocha capacidad de memoria.

STEP

Para el siguiente tema a tratar tendremos que echar mano del programa introducido anteriormente y denominado "Tabulador lanzamiento de cohetes"

Las líneas importantes de este punto son la 30, 40 y 70. Observará cuando ejecute el programa que aparecen en la pantalla los números del 10 al 1. La palabra STEP ("escalonamiento") en la línea 30, después del 1 controla esto. Cámbiese el −1 que sigue a STEP por −2 y vea lo que sucede. Si no se especifica el valor que acompaña a STEP,

el ordenador supone que se desea un STEP positivo de valor unidad, que es lo que ha ocurrido en los ejemplos anteriores.

La instrucción STEP se utiliza, pues, en los lazos FOR/NEXT para permitir al usuario especificar el valor del incremento (o decremento) de la variable de control. STEP no tiene por qué ser siempre un número entero aunque hay que asegurarse de que STEP sea negativo si el número que sigue a la palabra TO en la instrucción inicial FOR es inferior al que se halla delante de ella. Ensáyense los siguientes ejemplos:

```
10 FOR A=100 TO 1 STEP -12.5
20 PRINT TAB 8;A
30 NEXT A

        100
        87.5
        75
        62.5
        50
        37.5
        25
        12.5
```

```
10 FOR A=10 TO 1 STEP -0.175
20 PRINT TAB 8;A
30 NEXT A

        10
        9.825
        9.65
        9.475
        9.3
        9.125
        8.95
        8.775
        8.6
        8.425
        8.25
        8.075
        7.9
        7.725
        7.55
        7.375
```

En el bucle FOR/NEXT, STEP no tiene necesariamente que ser un número entero; puede ser una fracción, decimal, o el resultado de un cálculo y no tiene que alcanzar exactamente el valor límite del bucle. La secuencia se repite mientras el valor sea menor o igual al límite.

No puede cambiarse fácilmente el valor del STEP durante el curso de un bucle. Si ya se ha excedido el límite, el bucle será totalmente ignorado:

```
10 FOR F=1 TO 0
20 PRINT "X"
30 NEXT F
```

Podrá utilizarse esta idea para impedir bucles si se dan ciertas condiciones. Por ejemplo, si no se desease el trazado de una raya negra cuando X fuera igual a 6:

```
1000 FOR F=(X=6)*33 TO 31
1010 PRINT CHR$ 143;
1020 NEXT F
```

La prueba para averiguar si se ha excedido el valor límite se hace en la línea que contiene la instrucción FOR. Un interesante experimento es probar un valor 0 de STEP. Nunca se incrementa la variable de control y, por consiguiente, el bucle no termina nunca. Es posible saltar de los bucles FOR/NEXT sin problemas pero no se puede entrar en ellos a menos que se haya establecido la variable de control (efectivamente si se ha utilizado aquel ciclo con anterioridad). En un bucle FOR/NEXT se salta desde el NEXT hasta la línea siguiente a la de la instrucción FOR. Algunas versiones del BASIC permiten omitir la variable después de NEXT y entonces se incrementa la variable de control más reciente. Debe especificarse tal variable al ordenador.

GOSUB y RETURN

Una subrutina es una porción de programa, dentro de un programa mayor, que realiza una tarea específica. El programa principal se ejecuta línea por línea hasta que se hace una llamada a una subrutina

mediante la instrucción GOSUB. El ordenador apela al número especificado y empieza a trabajar por orden de instrucciones desde tal punto que llega a la palabra RETURN. Esta es la señal para volver al programa principal en la línea situada inmediatamente después de aquella a la que le envió a la subrutina.

La subrutina es útil si un determinado conjunto de cálculos tiene que realizarse un cierto número de veces en un programa y en distintos lugares del mismo. Por ejemplo, en un programa financiero puede haber un número de cálculos sobre impuestos en distintos lugares del programa. Cuando aparece la necesidad, el programa es enviado a la subrutina mediante el GOSUB y allí permanece hasta que encuentra la palabra RETURN, regresando entonces a la línea inmediatamente posterior a la de la instrucción GOSUB.

Una subrutina se escribe exactamente igual que el programa principal, con la particularidad de que se trata de un programa dentro de otro y está limitado por dos líneas, la del GOSUB y la del RETURN. La instrucción GOSUB se compone de un número de línea seguido de GOSUB y de otro número de línea. Ejemplo: 40 GOSUB 100 indica que hay un desplazamiento a la línea 100 y que se continúe la ejecución del programa ordenadamente a partir de dicha línea. Es lo mismo que si se hubiera dicho: 40 GO TO 100 ("Ir a"). Sin embargo, cuando se alcanza la línea que tiene la palabra RETURN, la acción revierte al programa principal en la línea que sigue a la que contiene la instrucción GOSUB (en este caso, el primer número de línea después de 40).

A continuación tenemos un sencillo ejemplo mostrando GOSUB y RETURN. Ejecútese varias veces para regresar al libro con objeto de tratar de este tema.

```
Su número es 234
234 al cuadrado es 54756

Su número es 23,76
23,76 al cuadrado es 564,5376

Su número es 4
4 al cuadrado es 16

Su número es 33
33 al cuadrado es 1089
```

```
  10 REM GOSUB/RETURN
  20 POKE 23609,100: REM INCLUIR
 PITIDO AL PULSAR TECLA
  30 INPUT "ENTRAR UN NUMERO ";A
  40 GO SUB 100
  50 GO TO 30
  90 REM SIGUE SUBRUTINA
 100 PRINT '"TU NUMERO ES ";A
 110 PRINT A;" AL CUADRADO ES ";
A↑2
 120 RETURN
```

La línea 30 pide un número, la 40 transfiere el control a la subrutina que empieza en la línea 100. Los cálculos requeridos son realizados y presentados los resultados, en la subrutina y después la línea 120 retorna el control a la línea inmediatamente posterior a la que envió el control a la subrutina, es decir, la 50. Como ésta tiene la instrucción GO TO ("Ir a"), la acción vuelve a la línea 30 que pide un nuevo número, y el ciclo vuelve a empezar otra vez.

Ejecute el siguiente programa que enfrenta a dos submarinos en una carrera. Con él veremos una subrutina haciendo algo un poco más interesante que lo del programa precedente.

```
  10 REM GOSUB EN UNA CARRERA
  15 PAPER 5: BORDER 5: CLS
  20 LET A$="▐
           ▄▟▙▄   "
  30 LET ORDENADOR=28
  40 LET HOMBRE=28
  50 LET X=5
  55 BEEP .01,RND*50
  60 GO SUB 100
  70 LET X=10
  75 BEEP .01,RND*50
  80 GO SUB 100
  90 GO TO 50
 100 IF X=5 THEN LET ORDENADOR=O
RDENADOR-RND: PRINT AT X,ORDENAD
OR; INK 6;A$: IF ORDENADOR<2 THE
N PRINT AT 0,0; PAPER 6; FLASH 1
; BRIGHT 1;"ORDENADOR GANA": STO
P
 110 IF X=10 THEN LET HOMBRE=HOM
BRE-RND: PRINT AT X,HOMBRE; INK
2;A$: IF HOMBRE<2 THEN PRINT AT
0,0; PAPER 6; FLASH 1;"HOMBRE GA
NA": STOP
 120 RETURN
```

Hay dos submarinos en la pantalla. El que está encima es el del ordenador y el de abajo es el suyo. Pulsando RUN y ENTER, el sub-

marino se desplaza a través de la pantalla, de derecha a izquierda. Cuando uno o el otro alcanza el margen, el programa se detiene presentando en pantalla el mensaje ORDENADOR GANA u HOMBRE GANA, según sea el caso.

Obsérvese que el submarino A$ se extiende en más de una línea. Siga pulsando la tecla SPACE una y otra vez cuando haya compuesto la parte del periscopio del submarino. Tenga en cuenta que hay un espacio *después* del extremo derecho del submarino. Resulta muy importante, como descubrirá si lo suprime.

Sonido

La instrucción BEEP del Spectrum puede utilizarse para dar más relieve a los programas, añadiendo sonidos apropiados cuando son destruidos los alienígenos, cuando rebotan las pelotas en la pared o si se logra vencer al ordenador en un juego de habilidad.

A primera vista pudiera parecer que un sonido de una sola voz, en un canal, carente de acompañamiento, que detiene las restantes acciones del ordenador cuando está en funciones, es bastante limitado. Puede, sin embargo, emplearse para añadir un sorprendente grado de interés a los programas.

Se dispone de la posibilidad de una pequeña extensión para altavoz en el Spectrum si se conecta en la clavija auricular (EAR). Aunque no aumenta mucho el volumen, se cuenta con una segunda fuente de sonido que puede mejorar la efectividad sonora del ordenador.

La instrucción BEEP (presionando ambas teclas SHIFT y mientras se mantiene la presión sobre la roja, se pulsa la de la letra Z) tiene dos parámetros (es decir, hay que poner dos números tras la palabra BEEP). El primero corresponde a la duración de la nota que se produce y el segundo expresa el tono.

Se tendrá idea de la clase de sonido que produce ejecutando la siguiente rutina de "música aleatoria":

```
 5 REM MUSICA ALEATORIA
10 BEEP RND/RND/3,RND*60-35
12 BORDER RND*7
15 BEEP RND/RND/2,RND*80-45
```

```
20 BORDER RND*7
25 BEEP RND/RND/3,RND*130-65
30 PAPER RND*7
40 CLS
45 BEEP RND/RND/2,RND*40-5
50 GO TO 10
```

El resultado de utilizar la instrucción BEEP en un bucle o en un par de ellos, puede ser muy interesante como se comprueba con la siguiente demostración:

```
  10 REM MUSICA EN BUCLES
  20 FOR A=-60 TO 60
  30 FOR B=.01 TO .03 STEP .01
  40 BEEP B,A: BEEP B,A/10*B: BE
EP B,ABS (A)
  50 NEXT B
  60 NEXT A
```

Los BEEPs producidos aleatoriamente suelen ser interesantes si se mantienen entre ciertos límites:

```
  10 LET TONO=INT (RND*24)-12
  20 LET DURACION=(INT (RND*8)+1
)/30
  30 BEEP DURACION,TONO
  40 IF RND>=.7 THEN GO TO 30
  50 GO TO 10
```

Como se hace constar en el Manual del Spectrum, ciertos números de tono producen determinadas notas musicales; por ejemplo, 0 para el DO central (C) y el 12 para el DO, una octava por encima, el −3 para dos notas por debajo del DO central. El programa siguiente convierte el Spectrum en un vibráfono en el que se utilizan las teclas de la hilera del fondo para producir las notas (en la clave de DO): DO (tecla Z), RE (tecla X), MI (tecla C), FA (tecla V), SOL (tecla B), LA (tecla N), SI (tecla M), DO (tecla K). Las notas continuarán produciéndose mientras se mantenga pulsada la tecla.

```
  10 REM PIANO
  20 LET A=CODE INKEY$
  30 IF A<66 OR A>90 THEN GO TO
20
  40 LET B=12*(A=75)+2*(A=88)+4*
(A=67)+5*(A=66)+7*(A=66)+9*(A=78
)+11*(A=77)
```

```
50 BEEP .04,B
60 IF INKEY$<>"" THEN GO TO 50
80 GO TO 20
```

Una vez se haya dominado esto se puede añadir un poco de color para acompañar a las notas con la variación que sigue (se ha agregado la línea 70 al programa precedente):

```
  10 REM PIANO
  20 LET A=CODE INKEY$
  30 IF A<66 OR A>90 THEN GO TO
20
  40 LET B=12*(A=75)+2*(A=88)+4*
(A=67)+5*(A=86)+7*(A=66)+9*(A=78
)+11*(A=77)
  50 BEEP .04,B
  60 IF INKEY$<>"" THEN GO TO 50
  70 BORDER B/2: PAPER B/2: CLS
  80 GO TO 20
```

Si se encuentra demasiado fatigoso tocar el vibráfono del Spectrum, se puede lograr que el ordenador lo haga por usted. Como se ve en la línea de datos (DATA), el "bien afinado Spectrum" utiliza la escala del DO natural.

```
  10 REM  ***  EL BIEN AFINADO
SPECTRUM  ***
  20 DIM A(8)
  30 FOR B=1 TO 8
  40 READ A(B)
  50 NEXT B
  70 LET B=INT (RND*8)+1
  75 LET M=(INT (RND*4)+1)/10
  80 BEEP M,A(B)
  85 IF RND>.9 THEN GO SUB 110
  90 GO TO 70
 100 DATA 0,2.039,3.86,4.98,7.02
,8.84,10.88,12
 110 LET Z=RND
 112 LET M=A(1)*(Z>=.5)+A(8)*(Z<
.5)
 115 BEEP 1,M
 120 PAUSE 25
 130 RETURN
```

Veremos muchos ejemplos de la utilización de la instrucción BEEP en los programas de este libro que proporcionarán ideas de sonidos que pueden añadirse a sus propios programas.

El último programa de esta sección es uno en el que el sonido es

un ingrediente importante y no solamente un acompañamiento. En esta variación del programa SIMON, el ordenador escoge un número entre uno y cuatro y lo sitúan en la pantalla con un sonido BEEP notable y un destello de color. Se tiene que pulsar el mismo número. El ordenador repetirá entonces el suyo añadiendo un segundo. A continuación usted ha de repetir ambos números en la secuencia correcta. Si el ordenador selecciona el mismo número dos veces seguidas, usted tiene que pulsar su tecla dos veces, soltando brevemente la tecla entre cada pulsación. Se gana el juego si es posible recordar correctamente una secuencia de siete cifras

```
   1 REM ** SIMON **
   5 LET A$=""
  10 FLASH 1
  20 PAPER 7
  30 FOR A=1 TO 7
  32 PRINT AT 10,10;"PREPARADO"
  35 BORDER RND*7
  40 LET A$=A$+STR$ (INT (RND*4)
+1)
  45 PAUSE 5
  50 NEXT A
  55 FLASH 0: CLS
  60 LET X=1
  70 FOR Q=1 TO X
  72 LET L=4*(CODE A$(Q)-48)
  73 LET T=VAL A$(Q)
  75 BEEP .05,10*T
  80 PRINT AT L,7; INK T;"████";"
 ";A$(Q);AT L+1,7;"████";AT L-1,7
;"████"
  85 BORDER RND*7
  90 PAUSE 20-X
 100 PRINT AT L,7; INK 6;"▚▚▚▚";A
T L+1,7;"▚▚▚▚";AT L-1,7;"▚▚▚▚"
 102 PAUSE 4
 105 CLS
 110 NEXT Q
 120 FOR B=1 TO X
 122 IF INKEY$<>"" THEN GO TO 12
2
 123 LET T$=INKEY$
 124 IF CODE T$=0 THEN GO TO 123
 125 CLS
 126 LET Y=4*(CODE T$-48)
 130 PRINT AT Y,7; INK Y/4;"████"
;T$;AT Y-1,7;"████";AT Y+1,7;"████
"

 145 BEEP .04,2.5*Y
 146 IF CODE T$<>CODE (A$(B)) TH
EN GO TO 300
 147 PAUSE 7
 148 CLS
 150 NEXT B
```

```
155 IF X=7 THEN PRINT "HAS GANA
DO!": BORDER RND*7: PAPER RND*7:
GO TO 155
160 LET X=X+1
165 PAUSE 50
170 GO TO 70
300 PRINT "PUNTUACION ";X-1
310 BORDER RND*7
320 PAPER RND*7
330 CLS
335 BEEP .02,RND*30
340 GO TO 300
```

Definición de funciones

Mediante DEF FN se pueden definir funciones en un programa y después utilizarlas cuando sean precisas para su ejecución. DEF FN permite ahorrar espacio y tiempo ya que cálculos complejos pueden ser definidos con un nombre corto y cuando son precisos se les recupera por ese nombre.

Hay cuatro cosas que definen la función:

— La palabra DEF.
— El nombre de la función que se compone de las letras FN seguidas por el nombre (una letra si se trata de una función numérica, una letra y el signo $ si es una función de cadena alfanumérica).
— El contenido de la función que sigue al nombre, entre paréntesis.
— La fórmula, utilizando el contenido, para el cálculo de la función.

Esto suena más complicado de lo que realmente es en la práctica. Observe este programa.

```
10 REM DEFINIR UNA FUNCION
20 DEF FN A(Z)=Z*Z
30 INPUT Z
40 PRINT Z,FN A(Z)
50 GO TO 30
```

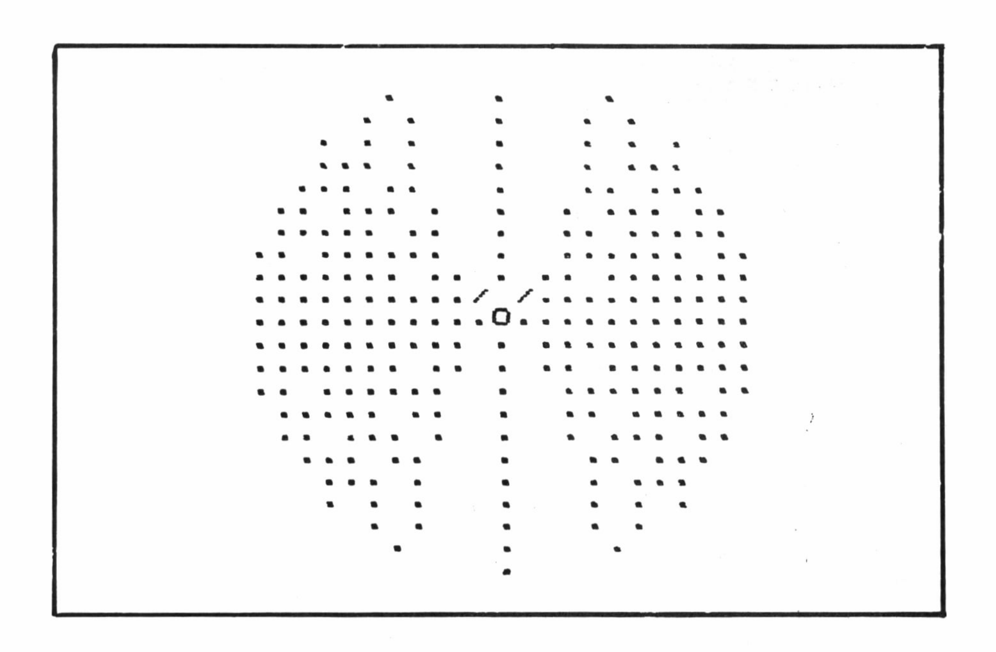
```
12                144
44                1936
```

```
23.76      564.5376
1111       1234321
44         1936
345        119025
4          16
11         121
```

La línea 20 define una función A, con el contenido Z que, en este caso, es Z^2. Por consiguiente, siempre que el programa llega a la expresión FNA(Z), calculará el cuadrado del valor asignado a la variable Z. Puede verse esto en la demostración.

En el siguiente programa, denominado MURCIELAGO, se define una función en la línea 70. La función produce la raíz cuadrada de la diferencia entre el cuadrado de dos variables, y en la rutina comprendida entre 120 y 210 utiliza el valor H (véase línea 130) para determinar las posiciones de presentación de los puntos que dibujarán el murciélago.

```
 10 REM MURCIELAGO
 20 REM DEMOSTRACION DEL DEF FN
 30 LET L=0
 40 LET P=10
 50 LET Q=17
 60 LET B=0
 70 DEF FN A(B)=SQR (L*L-B*B)
 80 PAPER 6: CLS
 90 INK 0: BORDER 2
100 PRINT AT P,Q;"O"
110 LET L=L+1
120 FOR B=0 TO L
130 LET H=FN A(B)
140 PRINT AT P+B,Q+H;"."
160 PRINT AT P+B,Q-H;"."
180 PRINT AT P-B,Q-H;"."
190 PRINT AT P-B,Q+H;"."
200 BEEP .02,50+H
210 NEXT B
220 IF L<11 THEN GO TO 110
230 PRINT INK 2;AT 9,16;"/ /"
```



La sentencia DIM y los conjuntos (arrays)

La sentencia DIM se utiliza para establecer una lista o relación a la que se pueda tener fácil acceso. En algunos programas es posible necesitar referirnos a elementos de una larga lista de números, como cuando se introduce (INPUT) una cantidad de datos (DATA) y deseamos utilizarlos de una cierta manera, como puede ser su presentación en la pantalla en un orden o magnitud.

Un conjunto (array) es una serie de espacios de memoria reservados en el ordenador y a los que referimos por el nombre del conjunto y por un subíndice. Para obtener conjunto que albergue tres elementos se introduce la sentencia DIM A(3) que proporciona espacio para un conjunto denominado A. Para albergar cuatro elementos se entraría DIM B(4).

Ejecute el programa siguiente que hará un poco más comprensible lo mencionado.

```
   10 REM ** DEMOSTRACION DE ORDE
NADORES **
   20 DIM B(4)
   30 FOR A=1 TO 4
   40 LET B(A)=INT (RND*9)+1
   50 NEXT A
   60 FOR A=1 TO 4
   70 PRINT TAB 6;"B(";A;") ES ";
B(A)
   80 NEXT A
```

```
          B(1) ES 6
          B(2) ES 9
          B(3) ES 7
          B(4) ES 1
```

Como se ve en la línea 20 el ordenador necesita que se dimensione un conjunto antes de poder utilizarlo. Esto se hace con la instrucción DIM, que se compone de un número de línea seguido por dicha palabra y el nombre del conjunto con su tamaño incluido entre paréntesis.

Los conjuntos de los que hemos estado hablando hasta ahora son unidimensionales, adecuados para hacer cosas como acoger una lista de números. Sin embargo, puede haber conjuntos que tengan más de una dimensión, que se denominan, con bastante razón, multi-dimensionales y que se establecen con la sentencia DIM y con más de un subíndice. Ejecútese el siguiente programa:

```
   10 REM CONJUNTOS MULTI-
   20 REM  DIMENSIONALES
   30 DIM A(4,4)
   40 FOR B=1 TO 4
   50 FOR C=1 TO 4
   60 LET A(B,C)=INT (RND*9)+1
   70 PRINT "A(";B;",";C;") ES ";
A(B,C)
   80 NEXT C
   90 NEXT B
  100 PRINT AT 16,15;"1 2 3 4"
  105 PRINT
  110 FOR B=1 TO 4
  120 PRINT TAB 13;B;TAB 15;A(B,1
);" ";A(B,2);" ";A(B,3);" ";A(B,
4)
  130 NEXT B
```

Cuando se ejecute se verá algo como esto:

```
A(1,1) ES 5
A(1,2) ES 7
A(1,3) ES 7
A(1,4) ES 6
A(2,1) ES 2
A(2,2) ES 9
A(2,3) ES 8
A(2,4) ES 5
A(3,1) ES 3
A(3,2) ES 3
A(3,3) ES 5
A(3,4) ES 7
A(4,1) ES 8
A(4,2) ES 9
A(4,3) ES 2
A(4,4) ES 2
```

	1	2	3	4
1	5	7	7	6
2	2	9	8	5
3	3	3	5	7
4	8	9	2	2

Los elementos de un conjunto estarán constituidos, en primer lugar, por números del uno al nueve que son presentados por la línea 70, de forma que puede verse lo que contiene cada elemento de dicho conjunto. La pequeña tabla que se presenta a su lado muestra cómo tales elementos están organizados. Cada uno de ellos es localizable en función de las coordenadas del conjunto. Siendo así, el elemento (3,3) se encuentra donde ambos números se intersectan, es decir, en el número 5. Observando la lista se comprueba realmente que el A(3,3) es igual al 5.

El dimensionado de los conjuntos consume memoria por lo que no debe hacerse mayor de lo necesario. El número de elementos en un conjunto es el producto de los dos valores comprendidos dentro del paréntesis; así, un conjunto o matriz A(4,4) tiene 16 elementos (4 × 4). En nuestro ejemplo puede comprobarse.

No hay razón para que no haya conjuntos con más de dos dimensiones, excepto por el hecho de que se hacen muy difíciles de manejar y el número de sus elementos aumenta alarmantemente. He aquí un programa para dimensionar y completar un conjunto tridimensional. Aunque se trata sólo de A(3,3,3) puede verse que el número de elementos es muy grande (3 × 3 × 3).

```
10 REM CONJUNTOS MULTI-
20 REM  DIMENSIONALES
30 DIM A(3,3,3)
40 FOR B=1 TO 3
```

```
 50 FOR C=1 TO 3
 55 FOR D=1 TO 3
 60 LET A(B,C,D)=INT (RND*9)+1
 70 PRINT "A(";B;",";C;",";D;")
ES ";A(B,C,D)
 75 NEXT D
 80 NEXT C
 90 NEXT B
```

```
A(1,2,3) ES 4
A(1,3,1) ES 8
A(1,3,2) ES 5
A(1,3,3) ES 7
A(2,1,1) ES 2
A(2,1,2) ES 2
A(2,1,3) ES 6
A(2,2,1) ES 2
A(2,2,2) ES 4
A(2,2,3) ES 6
A(2,3,1) ES 8
A(2,3,2) ES 5
A(2,3,3) ES 5
A(3,1,1) ES 9
A(3,1,2) ES 9
A(3,1,3) ES 9
A(3,2,1) ES 9
A(3,2,2) ES 2
A(3,2,3) ES 3
A(3,3,1) ES 1
A(3,3,2) ES 6
A(3,3,3) ES 8
```

Increméntese el número de dimensiones a cinco, como en nuestro ejemplo siguiente, y aunque se trata sólo de A(2,2,2,2,2), existen ahora 32 elementos (2^5).

```
A(1,1,1,1,1) ES 9
A(1,1,1,1,2) ES 4
A(1,1,1,2,1) ES 7
A(1,1,1,2,2) ES 2
A(1,1,2,1,1) ES 2
A(1,1,2,1,2) ES 9
A(1,1,2,2,1) ES 9
A(1,1,2,2,2) ES 9
A(1,2,1,1,1) ES 4
A(1,2,1,1,2) ES 6
A(1,2,1,2,1) ES 6
A(1,2,1,2,2) ES 5
A(1,2,2,1,1) ES 7
A(1,2,2,1,2) ES 8
A(1,2,2,2,1) ES 7
A(1,2,2,2,2) ES 8
A(2,1,1,1,1) ES 1
A(2,1,1,1,2) ES 7
```

```
A(2,1,1,2,1) ES 8
A(2,1,1,2,2) ES 4
A(2,1,2,1,1) ES 8
A(2,1,2,1,2) ES 2
```

```
  10 REM CONJUNTOS MULTI-
  20 REM  DIMENSIONALES
  30 DIM A(2,2,2,2,2)
  40 FOR B=1 TO 2
  50 FOR C=1 TO 2
  55 FOR D=1 TO 2
  56 FOR E=1 TO 2
  57 FOR F=1 TO 2
  60 LET A(B,C,D,E,F)=INT (RND*9
)+1
  70 PRINT "A(";B;",";C;",";D;",
";E;",";F;") ES ";A(B,C,D,E,F)
  72 NEXT F
  74 NEXT E
  75 NEXT D
  80 NEXT C
  90 NEXT B
```

Rompecódigos

He aquí el juego ROMPECODIGOS como demostración del empleo de un conjunto unidimensional. Es muy fácil jugar con este programa. El ordenador "piensa" en un número de cuatro dígitos y el jugador tiene diez oportunidades para averiguarlo. Una cifra correcta en posición equivocada produce un "blanco". Si la cifra y la posición son acertadas se produce un "negro".

```
  10 REM ROMPE/CODIGOS
  20 DIM C(4)
  30 DIM G(4)
  40 PRINT ''' PAPER 2; INK 6;"
ESTOY PENSANDO UN NUMERO       "
  50 PRINT PAPER 2; INK 6;" DE
CUATRO CIFRAS QUE DEBES        "
  60 PRINT PAPER 2; INK 6;" ADI
VINAR EN 10 INTENTOS.          "
```

```
  70 PRINT PAPER 2; INK 6;"  LAS
 CUATRO SON DIFERENTES         "
  80 PRINT PAPER 2; INK 6;"  PUL
SA UNA TECLA PARA EMPEZAR   "
  90 PAUSE 4E4
 100 CLS
 110 LET C(1)=INT (RND*9)+1
 120 FOR Z=2 TO 4
 130 LET C(Z)=INT (RND*9)+1: BEE
P .5,C(Z)
 140 FOR J=1 TO Z-1
 150 IF C(J)=C(Z) THEN GO TO 130
 160 NEXT J
 170 NEXT Z
 180 FOR G=1 TO 10: BEEP 1,2*G
 200 PRINT PAPER 1; INK 7;TAB 5;
"ENTRA TU INTENTO NUM. ";G;"
"
 210 INPUT A
 220 LET A1=A
 230 FOR Z=1 TO 4
 240 LET G(Z)=A-10*INT (A/10)
 250 LET A=INT (A/10)
 260 NEXT Z
 270 LET B=0
 280 LET W=0
 290 FOR Z=1 TO 4
 300 IF C(Z)<>G(Z) THEN GO TO
330
 310 LET B=B+1: BEEP .02,10*B
 320 LET G(Z)=0
 330 NEXT Z
 340 FOR Z=1 TO 4
 350 IF G(Z)=0 THEN GO TO 400
 360 FOR J=1 TO 4
 370 IF C(Z)<>G(J) THEN GO TO
390
 380 LET W=W+1: BEEP .02,7*W
 390 NEXT J
 400 NEXT Z
 420 PRINT INK 2;A1;" PUNTOS ";
INK 0;B;" NEGRO";
 430 IF B<>1 THEN PRINT "S";
 440 PRINT ", "; INK 2;W;" BLANC
O";
 450 IF W<>1 THEN PRINT "S";
 460 PRINT
 470 IF B=4 THEN PRINT " LO ADIV
INASTE EN ";G;" INTENTOS!"
 480 IF B<>4 THEN NEXT G
 490 PRINT " EL CODIGO ERA "; IN
K 2;C(4);C(3);C(2);C(1)
```

Conjuntos de cadenas o variables alfanuméricas

También puede trabajarse con variables alfanuméricas en forma similar a cómo se hace con las numéricas. Ejecute el siguiente programa para ver esto en la práctica; aplique cuatro palabras (cada una seguida de la instrucción ENTER).

```
10 REM CADENAS
20 DIM A$(4,10)
30 FOR B=1 TO 4
40 INPUT A$(B)
50 NEXT B
60 FOR B=1 TO 4
70 PRINT "A$(";B;") ES ";A$(B)
80 NEXT B
```

```
A$(1) ES AGUA
A$(2) ES RAZON
A$(3) ES DESECHO
A$(4) ES INGLATERRA
```

Aunque el segundo número de la instrucción DIM (en este caso 10) tiene que proporcionar capacidad para la cadena más larga que se pretenda introducir, sólo es necesario especificar el primero de tales elementos (como en la línea 70) para la presentación de toda la cadena.

Obsérvese que la diferencia principal entre un conjunto en cadena y uno numérico es el signo de dólar que sigue inmediatamente a la letra. Esto dice al ordenador que el nombre se refiere a una cadena alfanumérica.

He aquí un programa para ordenar alfabéticamente una cadena. Como se establece y como se demostró en el ejemplo anterior, el programa está adaptado para cinco palabras; si se desea que sean más, hay que cambiar el 5 de las líneas 20 y 40 por el número de palabras que haya que clasificar.

```
  10 REM CLASIFICACION DE CADENA
S
  20 DIM W$(5,10)
  30 LET B=0
```

```
  40 LET G=5
  50 FOR A=1 TO 5
  60 INPUT W$(A)
  70 PRINT W$(A)
  80 NEXT A: PRINT
  90 LET Z=1
 100 LET B=Z+1
 110 IF B>G THEN GO TO 190
 120 IF W$(B)>W$(Z) THEN GO TO 1
50
 130 LET Z=Z+1
 140 GO TO 100
 150 LET Q$=W$(Z)
 160 LET W$(Z)=W$(B)
 170 LET W$(B)=Q$
 180 GO TO 130
 190 PRINT W$(G)
 200 LET G=G-1
 210 IF G>0 THEN GO TO 90
```

```
REYERTA
RIGORISMO
REPIQUE
ROLAR
RADIO

RADIO
REPIQUE
REYERTA
RIGORISMO
ROLAR
```

El manejo de cadenas

Nuestro estudio sobre los conjuntos de cadenas (string arrays) nos conduce directamente a estas cadenas alfanuméricas y a su manejo. Como seguramente ya se sabrá, una cadena es un conjunto de caracteres alfanuméricos encerrados entre comillas (incluyendo símbolos y espacios, si se desea). Se asignan a una variable cuyo nombre es una letra seguida del signo de dólar. Con las cadenas se procede de la misma forma que con las variables numéricas, por medio de una instrucción de la forma LET A$ = "HOLA".

Existe un cierto número de funciones de estas cadenas que son muy útiles y que pueden utilizarse para su manipulación y para la extracción de partes de las mismas. Estas funciones son:

CODE X$ (Código)	Proporciona el código del primer carácter o letra de la cadena X$, de forma que si X$ se iguala a la palabra MICRO, por ejemplo, CODE X$ nos daría 77.
CHR$	Podemos comprobar si realmente 77 es el código de la primera letra de X$ (es decir, si es el código de M) pidiendo al ordenador que presente en pantalla el carácter 77 (PRINT CHR$ 77). Esto nos produce una M. En efecto, CHR$ es la función opuesta a CODE y convierte de nuevo el código en un carácter.
X$(TO 3)	Esto nos da una cadena que contiene los N caracteres de la izquierda de la variable alfanumérica X$. Así pues, X$(TO 3) nos dará "MIC".
LEN X$	Esta función nos da la longitud (número de caracteres) de una cadena; usando la que hemos establecido X$ de "MICRO", tendríamos en respuesta a LEN X$, de 5.
X$(n TO m) (n a m)	Esta función produce una cadena derivada de la X$ que incluye los caracteres comprendidos entre n y m. Empezando por el que ocupa el lugar n.X$(2 TO 4) nos da: "ICR".
X$(3 TO)	Es la opuesta, como podría esperarse, de la X$(a n), y nos da los caracteres de la derecha de la cadena. X$(3 TO) nos produce "CRO".
STR$ A	Convierte una variable (numérica) A en una cadena, así que si la variable era 234, la versión alfanumérica será "234". Esto no parece ser de mucha utilidad pero permite cierta manipulación con números cuando son cadenas lo que sería extremadamente difícil en su forma numérica. Volveremos sobre esta función con más detalle próximamente.
VAL X$	Es la opuesta de STR$ A y convierte el valor numérico de la cadena en un número. VAL X$, donde X$ es igual a "22 + 34", produciría 56.

A continuación se presenta un listado mostrando las funciones de cadenas en operación.

```
10 PRINT "LET X$='MICRO'"
20 LET X$="MICRO"
30 PRINT "CODE X$=";CODE X$
40 PRINT "CHR$ 77=";CHR$ 77
50 PRINT "X$(3 TO )=";X$(3 TO
)
60 PRINT "X$( TO 3)=";X$( TO 3
)
70 PRINT "LEN X$=";LEN X$
80 PRINT "X$(2 TO 4)=";X$(2 TO
4)
90 PRINT "LET X$='23+35'"
100 LET X$="23+35"
110 PRINT "VAL X$=";VAL X$
120 PRINT "LET X=34"
130 LET X=34
140 PRINT "LET X$=STR$ X"
150 LET X$=STR$ X
160 PRINT "X$=";X$
```

```
LET X$='MICRO'
CODE X$=77
CHR$ 77=M
X$(3 TO )=CRO
X$( TO 3)=MIC
LEN X$=5
X$(2 TO 4)=ICR
LET X$='23+35'
VAL X$=58
LET X=34
LET X$=STR$ X
X$=34
```

El empleo de la función LEN (longitud)

Si se desea presentar (PRINT) un cierto número de determinado carácter, por ejemplo, si se quiere dibujar una línea de trazos ("—") para hacer un subrayado, existen dos métodos. Por descontado, diferentes encabezamientos requieren distintas longitudes, por lo que se precisa saber cuántos caracteres hay que presentar. Si se trata de una cadena, como A$, se utiliza la función LEN para conocer la longitud de tal cadena A$, viniendo ésta expresada en el número de caracteres que hay que presentar.

```
10 FOR A=1 TO LEN A$
20 PRINT "-";
30 NEXT A
40 PRINT
```

La línea 40 desplaza la posición PRINT a la siguiente para continuar. Omítalo si no es necesario. El siguiente método es mucho más rápido y emplea sólo una línea de programa.

```
10 PRINT "-----------------------
-------------"( TO LEN A$)
```

La única desventaja es que se precisa especificar cuántos caracteres se requieren entre las comillas aunque nunca lleguen a presentarse. Es decir, se necesita conocer la longitud máxima que puede alcanzar A$ de forma que pueda ponerse tal número de caracteres en la constante de la cadena entre comillas después de PRINT.

El empleo de la función STR$ (string = cadena)

Esta función es muy útil y con frecuencia es olvidada. Como se dijo unas páginas antes, esta función convierte un número en su cadena equivalente tal como aparecería cuando se ordena su presentación en pantalla. Pruebe este programa:

```
10 PRINT 2
20 PRINT STR$ 2
30 PRINT 1/3
40 PRINT STR$ (1/3)
50 PRINT 9E15
60 PRINT STR$ 9E15
```

Se obtendrían estos resultados:

```
2
2
0.33333333
0.33333333
9E+15
9E+15
```

Podemos aprender mucho de estos ejemplos. La cadena que genera STR$ es la misma que se obtendría si con la instrucción PRINT se ordenara la presentación del número que constituye el contenido de STR$. Segundo, los números menores que 1 se asignan a una cadena con un 0 delante de la coma decimal, siempre que el primer dígito después de tal coma no sea cero (es decir, el número sea igual o mayor que 0,1 y menor que 1), y puede haber hasta ocho cifras decimales. También pueden ser menos si no se precisan todas. Es posible un total de hasta diez caracteres en toda la cadena. Sin embargo, si el número al que se aplica el enunciado STR$ tiene más de ocho cifras decimales se redondea en la octava. Ejemplo, STR$.3333333339 es "0,33333334". STR$ también es capaz de generar notación científica (del que, como se recordará, hemos tratado anteriormente), tal como 9E + 15. Obsérvese que, aunque el ordenador acepta 9E15, STR$ lo asigna como 9E + 15, es decir, que el exponente siempre lleva signo. Los números muy pequeños como el 0,000009 son asignados así: STR$ 0.00000009 es "9E − 7". Cuando se utiliza STR$, es aconsejable limitar los valores del número de forma que la función no empiece a utilizar la notación científica pues puede causar problemas.

Seguramente se estará pensando ahora; está bien..., ¿pero qué puede hacerse con esto?

La aplicación principal es convertir números en cadenas alfanuméricas, de forma que puedan utilizarse las facilidades del ordenador para manejar estas cadenas en la formación o redondeo de un determinado número de lugares decimales u otra finalidad por la que sea preciso determinar una cantidad, cifra por cifra. He aquí unos pocos ejemplos de aplicación de STR$.

(I) Alinéense las cantidades decimales. Supongamos que se dispone de una lista de números que hay que presentar en pantalla. Pruebe este programa:

```
10 LET A=RND*100
20 PRINT A
30 LET A=A*10
40 GO TO 20
```

Se obtendría algo parecido a esto:

```
8.581543
85.81543
858.1543
8581.543
85815.43
858154.3
8581543
85815430
8.581543E+8
8.581543E+9
8.581543E+10
8.581543E+11
```

Sería mucho más fácil de leer si se pudieran alinear las cifras decimales y esto es frecuentemente muy útil. Ensáyese esta rutina:

```
      79.025269
     790.25269
    7902.5269
   79025.269
  790252.69
 7902526.9
```

```
 10 LET A=RND*100
 20 PRINT TAB 15-LEN STR$ INT A
;A
 30 LET A=A*10
 40 GO TO 20
```

Produce un espaciado tal que los puntos decimales aparecen uno debajo del otro. Esto es adecuado para presentar una lista de números que se desea comparar rápidamente. ¿Puede verse cómo funciona el programa? Supongamos que el valor de A era 69,433594. Lo que hace el programa es tomar la parte entera de A (INT A, que es 69), la convierte en una cadena (STR$ INT A) y después mide su longitud (LEN STR$ INT A) que en este caso es 2. A continuación utiliza este número para determinar cuánto se ha de desplazar a la izquierda el comienzo de la presentación para el valor de A. Observe cómo se hace esto.

TAB 15 — LEN STR$ INT A

Esto significa que 15 es la localización en la que se va a situar la

coma decimal (punto decimal en el ordenador) y después cuenta hacia la izquierda el número de cifras enteras de la cadena (STR$ INT A).

(II) Hágase una presentación en pantalla con un cierto número de espacios decimales. Con frecuencia se precisa presentar, digamos, tres espacios decimales. Se recordará que el ejemplo anterior presentaba números con todas las cifras conocidas. Se puede utilizar STR$ para establecer cuántos números se van a presentar después de la coma (el punto en el ordenador). Considérese esta rutina:

```
       0.189
       1.893
      18.934
     189.346
    1893.463
```

```
  10 LET A=RND
  20 LET A$=STR$ A
  22 IF A$(1)="." THEN LET A$="-
" +A$
  25 LET B=LEN STR$ INT VAL A$
  27 PRINT TAB 15-B; (A$+("." AND
B=LEN A$)+"000") ( TO B+4)
  30 LET A=A*10
  40 GO TO 20
```

De esta forma se presentarán tres cifras decimales, añadiendo el 0 de delante y los que van detrás, si es preciso. Para hacer que se presenten Z cifras decimales, háganse los cambios siguientes en la línea 27; añádanse tantos espacios a A$ como números decimales se requieran (es decir, Z ceros) y si se hace la instrucción de división de la cadena (TO B + 1 + Z).

Sigue la explicación línea por línea:

La línea 10 establece el valor de A con el que empezar.

La línea 20 convierte A en una cadena.

La línea 22 añade un 0 delante del primer carácter si es un punto decimal. Desgraciadamente la función STR$ no es uniforme en su acción porque, a veces, proporciona un 0 delante de números menores que 1 y a veces no, según que el primer dígito después del punto

decimal sea 0 o no lo sea. Por consiguiente es una cuestión sencilla comprobar si se requiere un 0 o no. Si el primer carácter es un punto decimal, añádase un 0.

La línea 25 hace B igual al número de cifras de la parte entera de la cantidad que se va a presentar midiendo la longitud de esta parte en la cadena A$. Es necesario utilizar VAL A$ en lugar de A debido a que el ordenador puede encontrar una anomalía entre el último dígito del valor de A y el de A$ tal como lo ha establecido la función STR$, pues produciría un problema en este caso anómalo.

La línea 27 espacia la posición de la presentación (PRINT) como en el caso anterior (I), estableciendo después tres espacios decimales para la presentación de A$. En primer lugar, la cadena A$ se presenta completa y después se añade un punto decimal si tal cadena A$ ya representa un número entero y suficientes ceros para los tres lugares decimales. Pudiera preguntarse, ¿por qué añadir 4 a B si la presentación tiene tres decimales? Recuérdese que el punto decimal es un carácter extraordinario. Con el fin de la división, la parte delantera es tratada como una cadena larga siempre que esté comprendida entre comillas. Todo lo que se ha hecho es añadir caracteres para que a A$ le quepan los tres lugares decimales y después se han presentado estas tres cifras detrás del punto decimal. Esta rutina no redondea la tercera cifra decimal; existe otra en el programa SNIPPETS que sí lo hace.

(III) Ahorro de memoria. Con frecuencia es posible ahorrar capacidad de memoria utilizando cadenas para representar números en lugar de variables numéricas; tales cadenas se decodifican después mediante el enunciado VAL. Se puede poner el número en la variable alfanumérica utilizando STR$:

```
LET A$ = STR$(1024)
```

Y después se decodifica mediante el enunciado VAL:

```
PRINT VAL A$
```

Frecuentemente se encontrará que, de esta manera, se consume más memoria al convertir los números en cadenas que si hubieran sido entrados como variables numéricas pero, en ocasiones, este método hace maravillas.

Pruebe a aplicar VAL a una expresión como "ATN 1 * 4". Fun-

ciona, y esto es muy a menudo un recurso muy útil. Puede tenerse el número de una variable numérica entre comillas y, si previamente ha sido definido o asignado, será evaluado acertadamente. En realidad, VAL puede aplicarse a toda clase de expresiones numéricas y, en ocasiones, se emplea en lugar de la proposición DEF FN.

También es posible su utilidad si se desea generar números aleatorios varias veces en un programa. Al principio del programa póngase una instrucción LET A$ = "RND*6" y cada vez que se precise un número de esta clase se escribe LET R = VAL A$.

La función INKEY$

No se necesita pulsar el RETURN después de haberlo hecho sobre una tecla cuando se utiliza la función INKEY$, como el siguiente programa pondrá en claro.

Ensaye lo siguiente. Entre un número del 1 al 9 pulsando la tecla de tal número y verá en la pantalla USTED PULSO EL 6 ó USTED PULSO EL 1, etc. Presione la tecla 0 para terminar y en pantalla aparecerá USTED PULSO EL 0 y se detendrá el programa.

```
10 REM DEMOSTRACION DE INKEY$
20 PAUSE 100
40 LET A$=INKEY$
50 PRINT "HAS PULSADO EL ";A$
60 IF A$="0" THEN STOP
70 GO TO 20
```

```
HAS PULSADO EL 1
HAS PULSADO EL 5
HAS PULSADO EL 4
HAS PULSADO EL 3
HAS PULSADO EL 6
HAS PULSADO EL 2
HAS PULSADO EL 7
HAS PULSADO EL 7
HAS PULSADO EL 1
HAS PULSADO EL 7
HAS PULSADO EL 8
HAS PULSADO EL 2
HAS PULSADO EL 9
HAS PULSADO EL 3
HAS PULSADO EL 9
HAS PULSADO EL 1
```

```
HAS PULSADO EL 9
HAS PULSADO EL 8
HAS PULSADO EL 7
HAS PULSADO EL 0
```

El siguiente programa, denominado PREDICCION, también utiliza INKEY$. En este juego hay que tratar de adivinar el número (del 1 al 9) que va a seleccionar el ordenador. Este número se muestra en la pantalla, cerca de su centro, y el número situado en lugar inferior es la puntuación. Cuanto más baja sea la puntuación al final (es decir, cuando logre predecir con éxito el número del ordenador), tanto mejor. La pantalla permanecerá sin presentación hasta que se pulse una tecla. Las palabras "LA PUNTUACION ES" se presentarán en destellos.

```
TU NUMERO ES EL 4


EL MIO ES 6

LA PUNTUACION ES 1
```

```
  10 REM ** PREDICCION **
  20 LET E=8
  40 LET Q=0
  50 PAUSE 100
  70 LET A$=INKEY$
  80 PRINT AT 8,5; INK RND*7; PA
PER 9;"  TU NUMERO ES EL ";A$;"
"
  90 LET Q=Q+1: BEEP .02,Q*3
 110 LET W$=STR$ (INT (RND*9)+1)
 120 PRINT AT 12,E;; INK RND*7;
PAPER 9;"   EL MIO ES ";W$;"   "
 130 PRINT AT 14,5; INK RND*7; P
APER 9; FLASH 1; BRIGHT 1;"  LA
PUNTUACION ES ";Q;"  "
 135 STOP
 140 IF W$=A$ THEN BEEP .01,RND*
30: GO TO 130
 150 GO TO 50
```

Sigue el programa FABRICANTE DE ENREDOS que muestra igualmente a INKEY$ en acción. Utilizando las teclas de las letras "A", "Z", "K" y "M" hay que mover el signo $ desde la esquina inferior derecha hasta la superior izquierda sin cruzar ninguno de los pequeños cuadrados blancos. Obsérvese que ningún camino está garantizado

y que no existe mecanismo para evitar la trampa. Al final, el número de movimientos que se han necesitado se presenta en pantalla.

```
   10 REM FABRICANTE DE ENREDOS
   20 REM UTILIZA LAS LETRAS A Z
M K
   30 REM PARA MOVER EL SIGNO $
   40 REM DESDE LA ESQUINA
   50 REM INFERIOR DERECHA A LA
   60 REM SUPERIOR IZQUIERDA.
   70 BORDER 2: PAPER 7: CLS : BR
IGHT 1
   80 LET S=0
   90 FOR Y=1 TO 704
  100 LET T=INT (RND*3)
  110 PRINT INK (RND*5+1);("█" AN
D T=0)+(" " AND T<>0);
  120 NEXT Y
  130 PRINT AT 0,0;"   ";AT 1,0;"
  "
  140 LET X=30: LET Y=19
  150 LET M=X
  160 LET N=Y
  170 PRINT AT Y,X; INK 2;"$"
  180 LET S=S+1
  190 LET A$=INKEY$
  200 IF A$="" THEN GO TO 190
  210 IF A$="A" AND Y>0 THEN LET
Y=Y-1
  220 IF A$="Z" AND Y<20 THEN LET
 Y=Y+1
  230 IF A$="K" AND X<31 THEN LET
 X=X+1
  240 IF A$="M" AND X>0 THEN LET
X=X-1
  250 IF X=0 AND Y=0 THEN GO TO 2
80
  260 PRINT AT N,M;" "
  270 GO TO 150
  280 PRINT AT 10,10; FLASH 1; BR
IGHT 1;"LO LOGRASTE"
  290 PRINT '''TAB 4; FLASH 1; BR
IGHT 1;"HAS NECESITADO ";S;" SAL
TOS"
```

Obsérvese cómo en la línea 110 se hace uso de los métodos de evaluación de la lógica del ordenador. Esta línea asegura que si T (generada en la línea anterior) es 0 se presenta un cuadrado y si T es mayor que 0 se genera un espacio vacío. Esta es una forma eficaz, en relación con la conservación de memoria, de emular la proposición IF/THEN/ELSE (Si... Entonces... u Otra cosa) que existe en otros ordenadores y que se trata, junto con otras similares y con mayor detalle, en otra parte de este libro.

Si se desea un límite de tiempo para las respuestas del jugador, utilícese este método. Supongamos que sólo tiene unos pocos segundos para decidir si quiere seguir jugando o no. Si se es lento en tomar la decisión, el programa se detiene. Para la ejecución de esta rutina supongamos que el jugador tenía que pulsar la tecla de la R para una nueva jugada:

```
10 FOR F=1 TO 100
20 LET A$=INKEY$
30 IF A$="R" THEN GO TO 60
40 NEXT F
50 STOP
60 PRINT "VUELVE A JUGAR"
70 RUN
```

Alternativamente puede hacerse uso del contador de cuadros de presentación de imagen en pantalla para las entradas de tiempo o para otra cosa.

Para utilizar el contador de cuadros empléese esta rutina que establece primeramente el medidor de tiempo:

POKE 23673,255
POKE 23672,255

y para hacer uso de su valor en cualquier momento utilice:

LET T = (256*PEEK 23673 + PEEK 23672)/50

Esto proporcionará una medida en segundos bastante precisa si se presenta con PRINT T. Recuérdese que PAUSE y BEEP utilizan el contador de cuadros por lo que no puede ser empleada esta función si se usan dichas instrucciones en el programa. Para una medida en segundos es posible la rutina siguiente:

```
  10 POKE 23673,255
  20 POKE 23672,255
  30 LET T=(256*PEEK 23673+PEEK
23672)/50
  40 PRINT AT 0,0;T;"    "
  50 GO TO 30
```

Y ésta para un reloj digital:

```
10 REM RELOJ DIGITAL
20 BORDER 1: PAPER 2: CLS
```

```
  30 INK 7
  40 INPUT "ENTRA HORAS ";H
  50 INPUT "ENTRA MINUTOS ";M
  60 LET S=0
  70 POKE 23673,255
  80 POKE 23672,255
 110 PRINT AT 4,12;H;" ";
 120 IF M<10 THEN PRINT "0";
 130 PRINT M;" ";
 140 IF S<10 THEN PRINT "0";
 150 PRINT S;" "
 160 LET S=INT ((256*PEEK 23673+
PEEK 23672)/50)
 170 IF S>59.9999 THEN LET M=M+1
  POKE 23673,255: POKE 23672,255
 180 IF M=60 THEN LET H=H+1: LET
M=0
 190 IF H=13 THEN LET H=1
 200 GO TO 110
```

Si se desea más precisión en el tiempo sin tener que poner a 0 cada vez las direcciones 23673 y 23672 refiérase a las sección de PEEK y POKE que se verán más tarde en el libro.

READ/DATA/RESTORE

READ y DATA son formas muy convenientes de introducir información en un programa y su uso es relativamente fácil. Ejecútese el programa que sigue que muestra ambos READ y DATA en acción y vuelva después al libro para una explicación de cómo actúan.

```
  10 REM READ / DATA
  20 REM ****************
  30 REM LECTURA DE DATOS
  40 REM ****************
  50 DIM B(5)
  60 FOR A=1 TO 5
  70 READ B(A)
  80 NEXT A
  90 REM ****************
 100 REM VUELVA A LEER
 110 REM ****************
 120 FOR C=5 TO 1 STEP -1
 130 PRINT B(C)
 140 NEXT C
 150 DATA 13,35241,68,2,199999
```

```
199999
2
88
35241
13
```

En la línea 70 el ordenador llega a la instrucción READ (lectura)... Siempre que encuentra esta instrucción se dirige al primer elemento que sigue a la palabra DATA (Datos) y lo leer. En este caso es un conjunto (array). Los elementos que introduce DATA pueden estar en cualquier parte del programa (aunque es útil mantenerlos bastante cerca de la instrucción READ a la que están referidos).

Vuelva al programa "TABULADOR LANZAMIENTO DE COHETES" que utilizamos al principio del libro.

En el primer programa de esta sección, los datos (DATA) son números y están asignados a variables numéricas (los elementos del conjunto). En el programa sobre COHETES son cadenas (Línea 220) y se asignan a variables alfanuméricas A$ y después se presentan en la línea 120, a través del bucle denominado «cohete». Existe una tercera palabra que acompaña a READ y DATA: es RESTORE. Esta instrucción hace volver al principio de la lista de datos (DATA) y empezar la lectura (READ) nuevamente desde el primero. Aquí tenemos otro programa sencillo que muestra los datos en forma de cadenas e ilustra igualmente sobre la acción de la instrucción RESTORE.

```
  10 REM READ / DATA
  20 REM ****************
  30 REM LECTURA DE DATOS
  40 REM ****************
  50 DIM B$(21,5)
  60 FOR A=1 TO 21
  70 READ B$(A)
  75 IF 3*INT (A/3)=A THEN RESTO
RE
  80 NEXT A
  90 REM ****************
 100 REM VUELVA A LEER
 110 REM ****************
 120 FOR C=21 TO 1 STEP -1
 130 PRINT B$(C);"    ";
 140 NEXT C
 150 DATA "MOJAR","MATAR","MORIR
"

MORIR    MATAR    MOJAR    MORIR
MATAR    MOJAR    MORIR    MATAR
MOJAR    MORIR    MATAR    MOJAR
MORIR    MATAR    MOJAR    MORIR
MATAR    MOJAR    MORIR    MATAR
MOJAR
```

En este programa sólo hay tres datos por lo que la instrucción RESTORE actúa una vez se hayan leído los tres. La línea 80 asegura que esto ocurra cuando se completa la lectura de los tres mientras se efectúa el bucle A de 1 a 21. Obsérvese que los datos en cadena han de estar encerrados entre comillas.

Como se ha visto, se utiliza la instrucción READ en una línea para asignar valores a unas variables desde una secuencia de elementos contenidos en la instrucción DATA. Cada uno de dichos elementos **está separado del que sigue por una coma.** La instrucción READ se compone de un número de orden, seguido por la palabra READ y los nombres de las variables que han de ser asignados a las que se toman de la línea de datos (DATA).

Al llegar el programa a una instrucción READ se desplazará, como ya se ha dicho, a la primera DATA que encuentre, sin importar dónde se halle. El primer valor de DATA será asignado a la primera variable de la instrucción READ. Tras la asignación, el ordenador se desentiende de la instrucción DATA y la trata como si fuera del tipo REM (Información). Desplácese la línea 150 del listado anterior a la 25 y vuélvase a ejecutar el posterior programa nuevamente. Se verá que el ordenador ignora la línea 25 aunque la lee acertadamente.

Aunque los datos (DATA) se hallen dispersos por el programa, el ordenador los buscará como puede observarse en el programa que sigue:

```
   10 REM READ / DATA (LECTURA/DA
TOS"
   20 DATA "HOLA",7
   30 DIM B$(21,4)
   40 DIM Z(21)
   60 FOR A=1 TO 21
   70 READ B$(A),Z(A)
   80 IF 3*INT (A/3)=A THEN RESTO
RE
   85 DATA "PATO"
   90 NEXT A
   95 DATA 56,"PEPE"
  130 FOR C=1 TO 21
  140 PRINT B$(C),Z(C)
  145 DATA 22
  150 NEXT C
```

Es importante asegurarse de que hay suficientes datos para el número de veces que se desea que el ordenador lea. Elimínese la línea 145 del programa anterior y vuelva a ejecutarse. En la pantalla se presentará el mensaje de error "E OUT OF DATA,70:1" (faltan datos,70:1) porque el ordenador ha leído dos datos numéricos pero

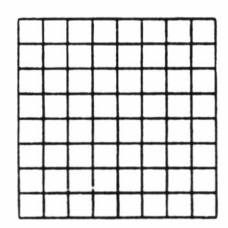
no ha podido encontrar el tercero y aún no ha llegado a la instrucción RESTORE.

Recuérdese que aunque no es esencial que los datos estén cerca de las líneas READ que los solicitan, los programas serán probablemente más fáciles de entender si se hallan de esta manera. También resulta más fácil saber las líneas que hay que alterar cuando se trabaja en un programa.

Gráficos definidos por el usuario

Una de las características más atractivas del ZX Spectrum es la facilidad para definir gráficos. Es fácil hacerlo y permite adaptar la parte visual de nuestros programas a nuestros deseos. En este capítulo desarrollaremos una forma muy simplificada del juego comercializado «Comecocos» para observar en acción los gráficos definidos por el usuario. A nuestro juego le llamaremos "DOTMAN" ("Hombre Punto") y consistirá en una sola criatura "Pacman" que trata de escapar de un "Ghost" (fantasma) mientras intenta comerse todos los puntos que pueda.

Los gráficos se definen sobre una cuadrícula de ocho por ocho, como la que sigue:

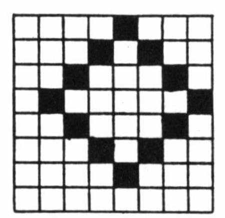


Podemos seleccionar cualquier tecla, desde la A a la U, e imponer nuestro gráfico con dicha tecla, de forma que cuando sea seleccionada, en lugar de presentar la letra correspondiente, trazará nuestro propio gráfico. Si bien ya sabemos que se perderán los gráficos si des-

conectamos el ordenador, no se verán, sin embargo, afectados por la instrucción NEW, por lo que es posible aplicar esta instrucción para entrar otro programa y seguir conservando los gráficos para un juego posterior.

Se cambia el contenido de los gráficos de una tecla mediante un bucle corto POKE. He aquí un ejemplo. Si se desea que aparezca una forma de diamante cada vez que se pulsan las teclas "CAP SHIFT" y "GRAPHICS", haciendo que la marca cursora en la pantalla sea una G, y accionando la letra A, procédase como sigue:

Primero, en su cuadrícula o parrilla (de las que existen varias al final de este Capítulo y que pueden fotocopiarse para una utilización futura) se dibuja el modelo que deseamos crear. Un diamante pudiera parecerse a esto:



El dibujo se introduce con la instrucción POKE mediante una serie de ocho instrucciones DATA en las que el 0 equivale a un cuadrado en blanco y el 1 a uno lleno. La línea superior de esta forma de diamante viene expresada por 00001000, un número binario de ocho dígitos. La segunda línea del diamante es 00010100. Compárense estas expresiones numéricas con los cuadrados negros del diagrama. Se indica al ordenador que el número utilizado es binario haciéndole preceder en el enunciado DATA con la instrucción BIN, que se obtiene con la tecla B. Después se presenta la rutina completa para la generación de la figura del diamante:

```
  10 REM DIAMANTE
  20 FOR J=0 TO 7
  30 READ Q
  40 POKE USR "A"+J,Q
  50 NEXT J
  60 PRINT "A"
  70 DATA BIN 00001000,BIN 00010
100,BIN 00100010,BIN 01000001,BI
N 00100010,BIN 00010100,BIN 0000
1000,BIN 00000000
```

Téngase en cuenta que la A de la línea 60 se pulsa en la modalidad de gráficos (cursor en G). Al ejecutar el programa, cambia incluso la A del listado: la línea 60 tiene esta representación:

```
60>PRINT "◇"
```

No hay duda de que hemos creado una forma de diamante bastante aceptable, como se ve en la representación que se obtiene:



Puede pensarse que los ceros delante de las cifras significativas no son necesarios; sin embargo, preferimos dejarlos ya que hacen mucho más fácil la comparación de los datos con la figura para el caso de que algo vaya mal. Aplíquese la siguiente instrucción DATA al programa anterior y véase lo que se consigue:

```
  70>DATA BIN 00110000,BIN 01110
000,BIN 01111110,BIN 01011111,BI
N 10011111,BIN 00010010,BIN 0001
0010,BIN 00000000
```

Se pretendía hacer un elefante y ha resultado bastante bien como puede verse en la pantalla.



Pasemos a la creación de nuestro juego "Comecocos", el "DOTMAN". Seguimos el proceso de conseguir un programa que funcione *antes* de definir los caracteres que forman parte de él. Esto asegura que lo importante es el propio programa, no los gráficos que se utilizan en el mismo. Es perfectamente posible ver la acción de un programa entero mediante el empleo de letras antes de decidir exactamente lo que va a representar cada una de ellas.

Nuestro juego será bastante sencillo. Sobre una parrilla de 14 por 14 existirá un "DOTMAN" sometido a nuestro control y un "FANTASMA" gobernado por el ordenador, así como una serie de puntos que

pueden ser engullidos por el DOTMAN. También habrá una especie de laberinto formado por muros que ni el FANTASMA ni el DOTMAN pueden franquear.

Al final del libro, en la sección denominada "Mejore sus programas", se sugiere que se esquematicen las partes principales del programa de forma que su estructura resulte clara, incluso antes de empezar a trabajar con él. Si se sigue el proceso que se describirá se verá lo útil que esto puede ser.

El programa "DOTMAN" ("comecocos")

La estructura para el DOTMAN es como sigue:

1-999 : Enviar la acción a las partes del programa para empezar.
1000-1999: El jugador controla el DOTMAN.
2000-3999: Movimiento del FANTASMA.
4000-4999: Fin del juego si el FANTASMA cae sobre el DOTMAN.
5000-5999: Definición del DOTMAN.
6000-6999: Definición del FANTASMA.
7000-7999: Presentación del laberinto.
8000-8999: Inicializar las variables.

Una vez se está ejecutando el programa, establecerá un bucle desde 1000 hasta 3999 (o cualquiera que sea el valor más alto que comprenda los movimientos del FANTASMA) una y otra vez, hasta que el FANTASMA alcance al DOTMAN, en cuyo momento el programa pasará a la línea 4000 para terminar la rutina del juego.

Escribamos la primera rutina de control:

```
10 GO SUB 8000
20 GO SUB 7000
30 GO SUB 5000
40 GO SUB 6000
```

Esto es todo lo que se necesita hasta este punto. Y ya que vamos a escribir todo el programa antes de definir el DOTMAN y el FAN-

TASMA, añadiremos 6999 RETURN y 5999 RETURN y seguiremos con la construcción del laberinto después que hayamos definido las variables. GA es la posición del FANTASMA en sentido transversal, GD es su coordenada descendente. EGA y EGD son las variables para "borrar" el FANTASMA cuando se mueve. En forma similar, DD y DA lo son para el DOTMAN en sentido transversal y descendente, y EDD con EDA para borrar.

Ponemos el laberinto y las rutinas para mover el DOTMAN y el FANTASMA, y resulta este programa:

```
  10 GO SUB 8000
  20 GO SUB 7000
  30 GO SUB 5000
  40 GO SUB 6000
1000 REM MOVIMIENTO DEL DOTMAN
1010 REM DOTMAN ES B,C,D,E EN LA
 MODALIDAD GRAPHICS
1020 PRINT AT EDD,EDA;" "
1030 PRINT AT DD,DA;A$
1035 LET EDD=DD: LET EDA=DA
1040 IF INKEY$="8" THEN IF DA<13
 THEN IF SCREEN$ (DD,DA+1)<>"X"
THEN LET DA=DA+1: LET A$="B"
1050 IF INKEY$="5" THEN IF DA>0
THEN IF SCREEN$ (DD,DA-1)<>"X" T
HEN LET DA=DA-1: LET A$="C"
1060 IF INKEY$="7" THEN IF DD>0
THEN IF SCREEN$ (DD-1,DA)<>"X" T
HEN LET DD=DD-1: LET A$="E"
1070 IF INKEY$="6" THEN IF DD<13
 THEN IF SCREEN$ (DD+1,DA)<>"X"
THEN LET DD=DD+1: LET A$="D"
1980 IF RND>.2 THEN GO TO 1000
1990 REM FANTASMA ES G EN
     GRAPHICS
2000 PRINT AT EGD,EGA;" "
2010 PRINT AT GD,GA;"G"
2015 LET EGD=GD: LET EGA=GA
2020 IF DD<GD THEN IF SCREEN$ (G
D-1,GA)<>"X" THEN LET GD=GD-1
2030 IF DD>GD THEN IF SCREEN$ (G
D+1,GA)<>"X" THEN LET GD=GD+1
2040 IF DA>GA THEN IF SCREEN$ (G
D,GA+1)<>"X" THEN LET GA=GA+1
2050 IF DA<GA THEN IF SCREEN$ (G
D,GA-1)<>"X" THEN LET GA=GA-1
2999 GO TO 1000
4999 STOP
5999 RETURN
6999 RETURN
7000 REM CONSTRUIR EL LABERINTO
7010 FOR G=1 TO 14
7020 FOR H=1 TO 14
7030 PRINT ".";
```

```
7040 NEXT H
7050 PRINT
7060 NEXT G
7070 PRINT AT 2,7;"XXXXX"
7080 PRINT AT 3,3;"X";AT 3,11;"X
"
7090 PRINT AT 4,3;"X";AT 4,11;"X
"
7100 PRINT AT 5,3;"XXXX";AT 5,9;
"XXX"
7110 PRINT AT 6,6;"X";AT 6,9;"X"
7120 PRINT AT 7,6;"X";AT 7,9;"X"
7130 PRINT AT 8,3;"X";AT 8,6;"X"
;AT 8,9;"X"
7140 PRINT AT 10,3;"X"
7150 PRINT AT 11,3;"XXXX"
7160 PRINT AT 12,9;"XXX"
7170 RETURN
8000 REM VARIABLES
8010 LET DA=0
8020 LET DD=0
8030 LET EDA=0
8040 LET EDD=0
8050 LET GA=13
8060 LET GD=13
8070 LET EGA=13
8080 LET EGD=13
8090 LET PUNTOS=0
8100 LET A$="B"
8999 RETURN
```

Si se ejecuta se verá que tenemos una versión del programa que funciona. Es un poco insulso de momento pero no debemos desanimarnos. Quedaremos agradablemente sorprendidos en lo bien que resultará cuando esté acabado. Puede mejorarse con facilidad, incluso en este punto al añadirle:

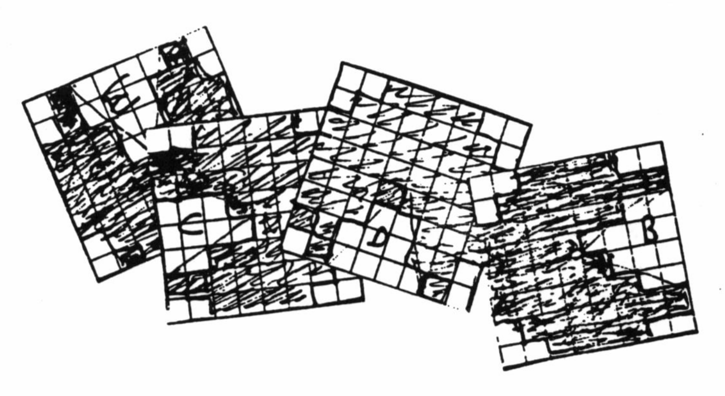
```
7065 INVERSE 1
7165 INVERSE 0
```

La instrucción INVERSE funciona como el BRIGHT y el FLASH; 1 significa activado y 0 desactivado. Vuélvase a ejecutar el programa y obsérvese el efecto del INVERSE sobre las paredes del laberinto. Las realza inmediatamente, ¿verdad? Debe haber cuatro DOTMANS, uno en cada dirección, de forma que su boca apunte en la dirección en que se mueva. Por ello la variable A$, que es el DOTMAN, se cambia al final de las líneas 1040 a 1070.

Añádase lo siguiente para definir los DOTMANS:

```
5000 REM DEFINICION DE LOS
DOTMANS
5010 FOR J=0 TO 7
5020 READ Q
5030 POKE USR "ᗧ"+J,Q
5040 NEXT J
5050 DATA BIN 00111100,BIN 01111
111,BIN 11111100,BIN 11110000,BI
N 11111000,BIN 11111100,BIN 0111
1111,BIN 00111100
5060 FOR J=0 TO 7
5070 READ Q
5080 POKE USR "ᗤ"+J,Q
5090 NEXT J
5100 DATA BIN 00111100,BIN 11111
110,BIN 00111111,BIN 00011111,BI
N 00001111,BIN 00111111,BIN 1111
1110,BIN 00111100
5110 FOR J=0 TO 7
5120 READ Q
5130 POKE USR "ᗣ"+J,Q
5140 NEXT J
5150 DATA BIN 00111100,BIN 01111
110,BIN 11111111,BIN 11111111,BI
N 11110111,BIN 11100111,BIN 0100
0010,BIN 01000010
5160 FOR J=0 TO 7
5170 READ Q
5180 POKE USR "ᗢ"+J,Q
5190 NEXT J
5200 DATA BIN 01000010,BIN 01000
010,BIN 11100111,BIN 11101111,BI
N 11111111,BIN 11111111,BIN 0111
1110,BIN 00111100
```

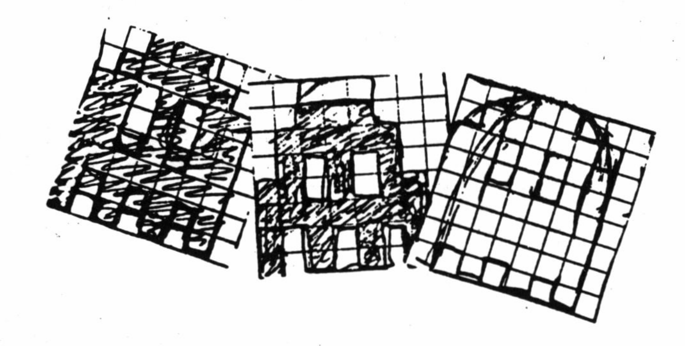
Hemos esbozado un número de posibles figuras para el DOTMAN antes de decidirnos por la marcada con la B. A este dibujo se le ha dado la vuelta para conseguir los números binarios para las cuatro figuras.



```
= GRAFICO DE B
= GRAFICO DE C
= GRAFICO DE D
= GRAFICO DE E
```

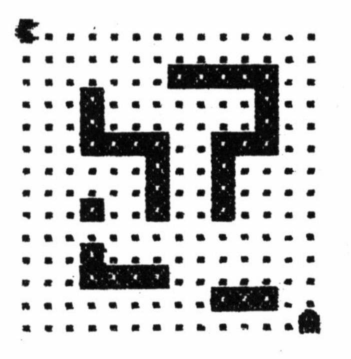
Agréguese algún color insertando INK 2 en la línea 1030. Esto hará el DOTMAN de color rojo. INK 1 en la línea 2010 pondrá azul al FANTASMA cuando esté completo.

He aquí nuestros bocetos de trabajo para el FANTASMA.

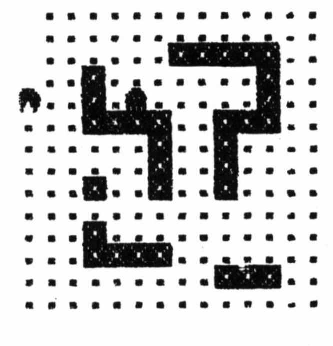


El FANTASMA que finalmente conseguimos es un pariente próximo del de la izquierda del dibujo. Aunque no parece particularmente prometedor aquí, resulta bastante bien en la pantalla. Probablemente ocurrirá que, de vez en cuando, se deseará modificar el gráfico final cuando se vea en la pantalla, y resultará bastante fácil trabajar directamente desde ella, sabiendo pronto qué elementos de la instrucción DATA (de datos) hay que cambiar. De cualquier manera, he aquí nuestra versión final del DOTMAN, completa con unas cuantas presentaciones en la pantalla. Como ocurre con frecuencia estas presentaciones no hacen justicia a la apariencia del juego. Seguramente que se disfrutará jugándolo e imaginando ideas para mejorarlo y hacerlo cada vez más parecido al juego comercializado "COMECOCOS".

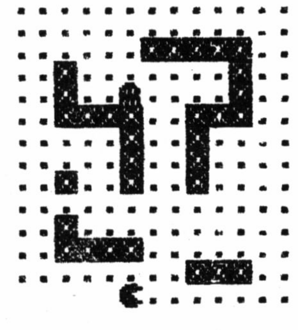
Un cambio que pudiera desearse es añadir una rutina para generar un laberinto aleatorio (poniendo X,s al azar sobre la parrilla en la subrutina 7000). Asimismo se puede hacer que los DOTMANS y el fantasma empiecen en posiciones aleatorias dentro del laberinto. Una característica interesante pudiera ser la posibilidad de marcar la puntuación más alta obtenida.



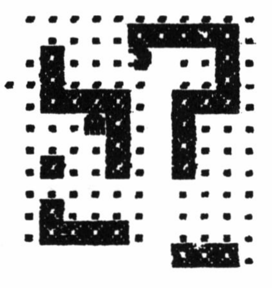
0 PUNTOS



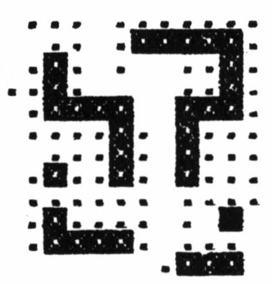
9426 PUNTOS



44783 PUNTOS



188560 PUNTOS



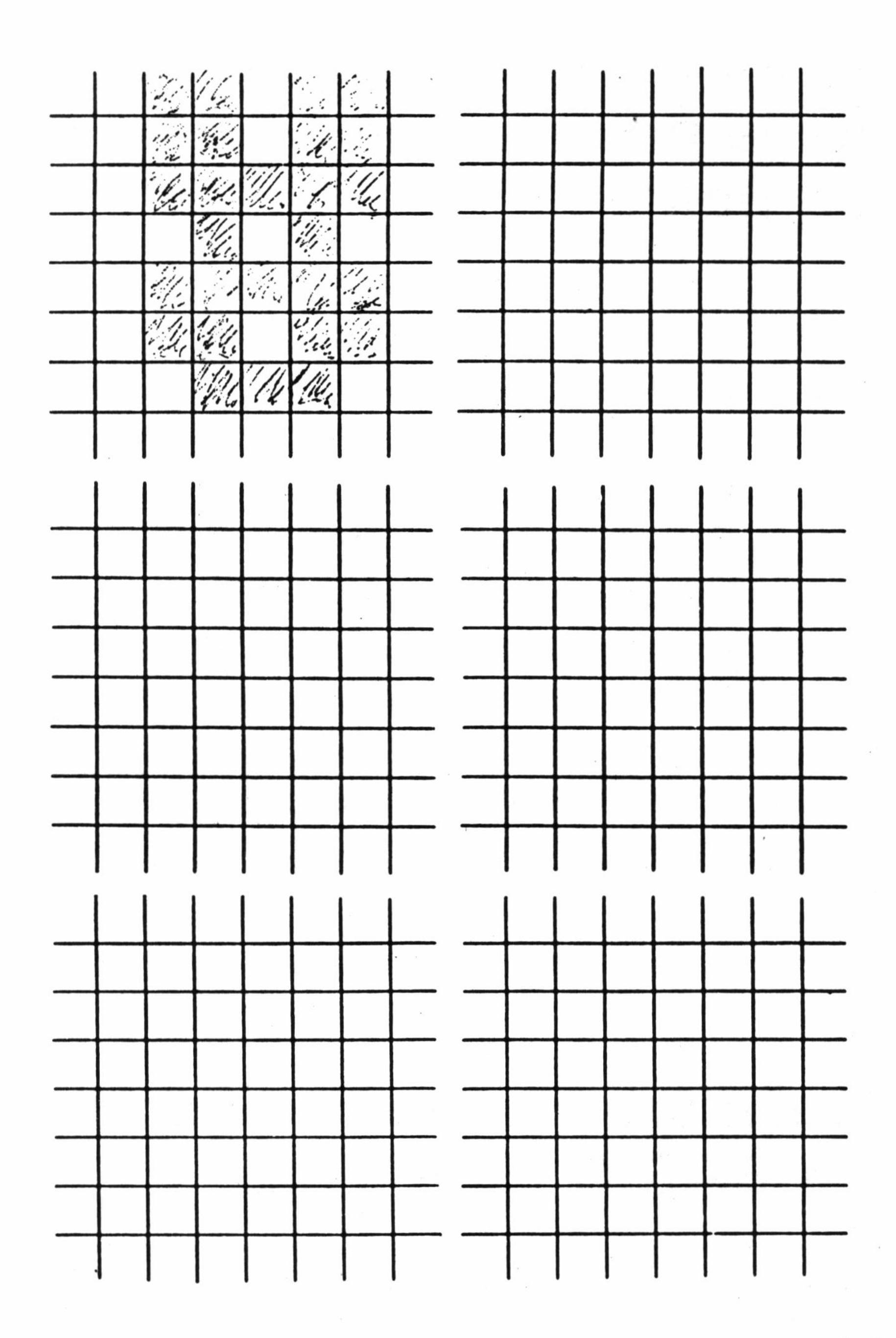
214487 PUNTOS

FIN DEL JUEGO

```
  10 GO SUB 8000
  20 GO SUB 7000
  30 GO SUB 5000
  40 GO SUB 6000
1000 REM MOVIMIENTO DEL DOTMAN
1010 REM DOTMAN ES B,C,D,E EN LA
 MODALIDAD GRAPHICS
1020 PRINT AT EDD,EDA;" "
1025 IF SCREEN$ (DD,DA)="." THEN
 LET PUNTOS=PUNTOS+2357
1030 PRINT AT DD,DA; INK 2;A$
1032 IF GA=DA AND GD=DD THEN GO
TO 4000
1035 LET EDD=DD: LET EDA=DA
1040 IF INKEY$="8" THEN IF DA<13
 THEN IF SCREEN$ (DD,DA+1)<>"X"
THEN LET DA=DA+1: LET A$="ᗧ"
1050 IF INKEY$="5" THEN IF DA>0
THEN IF SCREEN$ (DD,DA-1)<>"X" T
HEN LET DA=DA-1: LET A$="ᗤ"
```

```
1060 IF INKEY$="7" THEN IF DD>0
THEN IF SCREEN$ (DD-1,DA)<>"X" T
HEN LET DD=DD-1: LET A$="ᗢ"
1070 IF INKEY$="6" THEN IF DD<13
 THEN IF SCREEN$ (DD+1,DA)<>"X"
THEN LET DD=DD+1: LET A$="ᗣ"
1980 IF RND>.2 THEN GO TO 1000
1985 PRINT AT 7,16; FLASH 1;PUNT
OS;" PUNTOS"
1990 REM FANTASMA ES G EN
     GRAPHICS
2000 IF SCREEN$ (GD,GA)=" " THEN
 PRINT AT EGD,EGA;" ": GO TO 201
0
2005 PRINT AT EGD,EGA; INK RND*6
;"."
2010 PRINT AT GD,GA; INK 1;"ᗣ"
2012 IF GA=DA AND GD=DD THEN GO
TO 4000
2015 LET EGD=GD: LET EGA=GA
2020 IF DD<GD THEN IF SCREEN$ (G
D-1,GA)<>"X" THEN LET GD=GD-1
2030 IF DD>GD THEN IF SCREEN$ (G
D+1,GA)<>"X" THEN LET GD=GD+1
2040 IF DA>GA THEN IF SCREEN$ (G
D,GA+1)<>"X" THEN LET GA=GA+1
2050 IF DA<GA THEN IF SCREEN$ (G
D,GA-1)<>"X" THEN LET GA=GA-1
2999 GO TO 1000
4000 PRINT AT 17,0; INK RND*6; P
APER 9; FLASH 1;"FIN DEL JUEGO"
4010 BEEP .01,RND*60
4020 PRINT INK RND*6; FLASH 1;AT
 EGD,EGA;" ";AT GD,GA;"■"
4030 PRINT INK RND*6; FLASH 1;AT
 EDD,EDA;" ";AT DD,DA;"■"
4040 GO TO 4000
5000 REM DEFINICION DE LOS
DOTMANS
5010 FOR J=0 TO 7
5020 READ Q
5030 POKE USR "ᗤ"+J,Q
5040 NEXT J
5050 DATA BIN 00111100,BIN 01111
111,BIN 11111100,BIN 11110000,BI
N 11111000,BIN 11111100,BIN 0111
1111,BIN 00111100
5060 FOR J=0 TO 7
5070 READ Q
5080 POKE USR "ᗧ"+J,Q
5090 NEXT J
5100 DATA BIN 00111100,BIN 11111
110,BIN 00111111,BIN 00011111,BI
N 00001111,BIN 00111111,BIN 1111
1110,BIN 00111100
5110 FOR J=0 TO 7
5120 READ Q
5130 POKE USR "ᗣ"+J,Q
5140 NEXT J
```

```
5150 DATA BIN 00111100,BIN 01111110,BIN 11111111,BIN 11111111,BIN 11110111,BIN 11100111,BIN 01000010,BIN 01000010
5160 FOR J=0 TO 7
5170 READ Q
5180 POKE USR "■"+J,Q
5190 NEXT J
5200 DATA BIN 01000010,BIN 01000010,BIN 11100111,BIN 11101111,BIN 11111111,BIN 11111111,BIN 01111110,BIN 00111100
5999 RETURN
6000 REM DEFINICION DEL FANTASMA
6010 FOR J=0 TO 7
6020 READ Q
6030 POKE USR "■"+J,Q
6040 NEXT J
6050 DATA BIN 00111000,BIN 01111100,BIN 11010110,BIN 11010110,BIN 11111110,BIN 11111110,BIN 10101010,BIN 10101010
6999 RETURN
7000 REM CONSTRUIR EL LABERINTO
7010 FOR G=1 TO 14
7020 FOR H=1 TO 14
7030 PRINT INK RND*6;".";
7040 NEXT H
7050 PRINT
7060 NEXT G
7065 INVERSE 1
7070 PRINT AT 2,7;"XXXXX"
7080 PRINT AT 3,3;"X";AT 3,11;"X"
7090 PRINT AT 4,3;"X";AT 4,11;"X"
7100 PRINT AT 5,3;"XXXX";AT 5,9;"XXX"
7110 PRINT AT 6,6;"X";AT 6,9;"X"
7120 PRINT AT 7,6;"X";AT 7,9;"X"
7130 PRINT AT 8,3;"X";AT 8,6;"X";AT 8,9;"X"
7140 PRINT AT 10,3;"X"
7150 PRINT AT 11,3;"XXXX"
7160 PRINT AT 12,9;"XXX"
7165 INVERSE 0
7170 RETURN
8000 REM VARIABLES
8010 LET DA=0
8020 LET DD=0
8030 LET EDA=0
8040 LET EDD=0
8050 LET GA=13
8060 LET GD=13
8070 LET EGA=13
8080 LET EGD=13
8090 LET PUNTOS=0
8100 LET A$="■"
8999 RETURN
```



Eliminación de una parte de la presentación en pantalla

Se trata de una subrutina útil que permite eliminar cualquier número de líneas del fondo de la pantalla. Pueden conservarse algunas líneas de instrucciones en la parte superior de la presentación durante el juego o bien la puntuación lograda u otras instrucciones especiales, eliminando el resto. La subrutina debe denominarse "GOSUB 8010".

```
8010 INPUT "CUANTAS LINEAS HAY
QUE DESPEJAR? ";C
8020 IF C<0 OR C>21 THEN GO TO
8010
8030 FOR F=21 TO 21-C STEP -1
8040 PRINT AT F,0;"
                                "
8050 NEXT F
8060 PRINT AT F+1,0;
8070 RETURN
```

La línea 8010 pide las líneas que hay que eliminar, empezando desde la 21 en el fondo de la pantalla hacia arriba. Resulta impresionante. La instrucción INPUT (Entrada) no es un enunciado a toda prueba; posiblemente le gustará experimentar con él por sí mismo. La instrucción de la línea 8060 desplaza la posición de la presentación (PRINT) al principio de la parte de la pantalla que se acaba de despejar.

Desplazamiento de pantalla en BASIC (Scrolling)

Los programas en código máquina pueden pasarse fácilmente en grandes porciones de memoria de carga en bloque desde una

localización a otra para permitir, por ejemplo, que la pantalla se cargue en ciertas direcciones de memoria. En general el lenguaje BASIC es normalmente demasiado lento para permitir hacer esto. El único método que puede prácticamente utilizarse para desplazar la imagen de la pantalla es almacenarla en un conjunto (array) en cadena y después presentarla con la instrucción PRINT. A continuación se dan cuatro ejemplos de programas que permiten que toda la imagen en pantalla se desplace hacia arriba, hacia abajo, izquierda o derecha.

Estas rutinas son apreciablemente más lentas que la instrucción SCROLL, especialmente cuando se trata de desplazamientos a izquierda y derecha ya que se presentan de línea en línea.

Desplazamiento ascendente

```
   5 REM DESPLAZAMIENTO ASCENDEN
TE
  10 DIM A$(704)
  20 INPUT A$
  30 PRINT AT 0,0;A$
  40 LET A$=A$(33 TO )+"
                                 "
  50 GO TO 30
```

Desplazamiento descendente

```
   5 REM  DESPLAZAMIENTO  DESCEN
DENTE
  10 DIM A$(704)
  20 INPUT A$
  30 PRINT AT 0,0;A$
  40 LET A$="
                  "+A$( TO 672)
  50 GO TO 30
```

Desplazamiento hacia la izquierda

```
   5 REM  DESPLAZAMIENTO  A  LA
IZQUIERDA
  10 DIM A$(704)
  20 INPUT A$
  30 PRINT AT 0,0;A$
  40 FOR F=1 TO 673 STEP 32
  50 LET A$(F TO F+31)=A$(F+1 TO
 F+31)+" "
  60 NEXT F
  70 GO TO 30
```

Desplazamiento hacia la derecha

```
   5 REM  DESPLAZAMIENTO  A  LA
DERECHA
  10 DIM A$(704)
  20 INPUT A$
  30 PRINT AT 0,0;A$
  40 FOR F=1 TO 673 STEP 32
  50 LET A$(F TO F+31)=" "+A$(F
TO F+30)
  60 NEXT F
  70 GO TO 30
```

Conservación de líneas en los márgenes de la pantalla

La forma más fácil de lograr esto es mediante el uso de la posibilidad de dividir las cadenas alfanuméricas para desplazar ("SCROLL")

sólo ciertas partes de ellas. Veamos primeramente el desplazamiento vertical ascendente. Sea L el número de líneas que no se desplazan.

```
  10 DIM A$(704)
  20 INPUT A$
  25 INPUT L
  30 PRINT AT 0,0;A$
  40 LET A$(L*32+1 TO ) =A$((L+1)
*32+1 TO )
  50 GO TO 30
```

Sigue el desplazamiento vertical descendente. La variable L es el número de líneas no desplazables en el fondo de la pantalla.

```
  10 DIM A$(704)
  20 INPUT A$
  25 INPUT L
  30 PRINT AT 0,0;A$
  40 LET A$( TO 704-L*32)="
                                  "+A$(
TO 704-(L+1)*32)
  50 GO TO 30
```

Es posible aplicar la misma técnica para los desplazamientos laterales, pero debido a que éstos son bastante lentos, no merecen la pena. Puede extenderse esta idea para permitir que líneas de cualquier parte de la pantalla se mantengan estacionarias mientras que las que están encima o debajo se desplazan; ello se haría modificando la división de la cadena pero si se requiere un cálculo complejo sería demasiado lenta la rutina.

Otra técnica aplicable a estas rutinas es la de "fuga y entrada", por la que lo que desaparece por un lado reaparece por el opuesto. He aquí cómo hacerlo con un desplazamiento vertical:

```
  10 DIM A$(704)
  20 INPUT A$
  30 PRINT AT 0,0;A$
  40 LET A$=A$(33 TO )+A$( TO 32
)
  50 GO TO 30
```

Y un desplazamiento vertical descendente con "fuga y entrada":

```
10 DIM A$(704)
20 INPUT A$
```

```
  30 PRINT AT 0,0;A$
  40 LET A$=A$(673 TO )+A$( TO 6
72)
  50 GO TO 30
```

Desplazamiento hacia la izquierda con "fuga y entrada".

```
  10 DIM A$(704)
  20 INPUT A$
  30 PRINT AT 0,0;A$
  40 FOR F=1 TO 673 STEP 32
  50 LET A$(F TO F+31)=A$(F+1 TO
F+31)+A$(F)
  60 NEXT F
  70 GO TO 30
```

Para el desplazamiento a la derecha con "fuga y entrada", cámbiese la línea 50 por la que sigue:

```
  50>LET A$(F TO F+31)=A$(F+31)+
A$(F TO F+30)
```

Finalmente, recuérdese: en lugar de utilizar PRINT para situar algo en pantalla empléese: LET A$(X) = "X", puesto que lo que se hace presentar mediante la instrucción PRINT no se desplaza, sólo aparece en pantalla el contenido del conjunto (array) A$.

Movimiento de gráficos

La teoría para estos movimientos se elabora sobre la base de que primero se dibuja un carácter en una posición durante un tiempo breve, después se borra y vuelve a presentarse en otra posición. Se utilizan variables para recordar la posición de la representación. Veamos un programa ejemplo que elaboramos según esta "teoría".

```
  10 LET X=0
  20 PRINT AT 5,X;"█"
  30 PRINT AT 5,X;" "
  40 LET X=X+1
  50 GO TO 20
```

Este no es un programa muy bueno, el cuadrado negro presentado parece detellear al moverse en la pantalla y el programa se detiene con un mensaje de error cuando el cuadrado negro alcanza el extremo derecho de la pantalla. Lo que ha ocurrido es que X es una variable que dice al ordenador dónde tiene que presentar el cuadrado negro, y si el valor correspondiente es mayor que 31 ya no se puede presentar ya que la pantalla se extiende desde el 1 al 31, y cualquier intento de aplicar un número superior a 31 haría la presentación del cuadrado fuera de ella. Siendo el Spectrum una pequeña máquina inteligente decide que esto no puede ocurrir, por lo que detiene el programa y dice que hay un error para que pueda corregirse. Hagámoslo. La forma más fácil es disponer que si el valor de X cae fuera del alcance permitido cambie automáticamente el programa a otro más adecuado. Esto puede hacerse añadiendo una línea como la siguiente:

```
45 IF X>31 THEN LET X=0
```

Pero dado que ya tenemos una línea que dice LET X = 0, podría conseguirse lo mismo remitiendo el programa a dicha línea, la 10.

```
45 IF X>31 THEN GO TO 10
```

Esta técnica se usa mucho en los programas, reenviando al principio para volver a establecer ciertas variables en sus valores iniciales. En este ejemplo, ambos métodos consiguen los mismos resultados, aunque es posible encontrar programas en los que sólo uno de ellos es conveniente, o uno es mejor que el otro.

El paso siguiente tiene por objeto mejorar la presentación. Una forma de hacerlo es asegurarse que la presentación dura más que el borrado. Pruébese esto:

```
10 LET X=0
20 PRINT AT 5,X;"■"
30 FOR F=1 TO 10
40 NEXT F
45 IF X>31 THEN GO TO 10
50 PRINT AT 5,X;" "
60 LET X=X+1
70 IF X>31 THEN LET X=0
80 GO TO 20
```

Parece funcionar bastante bien pero en muchos programas hay que realizar otros cálculos que originarán retardos, y el bucle de pérdida de

tiempo de las líneas 30 y 40 retrasará las cosas innecesariamente por lo que no es una solución muy conveniente. Tratemos de reducir la sensación de destelleo en el movimiento haciendo menor el tiempo entre el espacio donde se va a hacer la presentación y la propia presentación. Ensaye lo que sigue:

```
10 LET X=0
20 LET P=X
30 LET X=X+1
40 IF X>31 THEN LET X=0
50 PRINT AT 5,P;" ";AT 5,X;"■"
60 GO TO 20
```

Observe cómo el desplazamiento es más regular.

Aquí, X es la variable que recuerda la posición donde se va a hacer la presentación y P la de la posición previa, de forma que pueda borrarse al disponer un espacio vacío sobre ella. La línea 10 hace que X empiece con un valor 0. La línea 20 determina el de P haciéndolo igual a X *antes* de que X se incremente en valor en la línea 30. El incremento puede tener cualquier valor; pruébese a cambiar el 1 que sigue al signo + y véase el efecto. Da la sensación de que el movimiento es más rápido y esto puede ser ventajoso o desventajoso. **Póngase nuevamente el 1 después del +.**

La acción de la línea 40 es asegurar que cuando la representación del cuadrado ha alcanzado el lado derecho de la pantalla, vuelve al izquierdo al restablecerse el valor 0 a X. Ello nos proporciona una provisión constante de cuadraditos. La línea 50 hace toda la presentación; obsérvese cómo dos funciones AT pueden ponerse en la misma línea unidas por medio de un punto y coma. Lo mismo puede hacerse con TAB, incidentalmente. Pruebe a poner las funciones AT en dos líneas separadas para ver si se produce alguna diferencia en el programa:

```
50>PRINT AT 5,P;" "
55 PRINT AT 5,X;"■"
```

Se puede acortar el programa en una línea haciendo que la 30 sea **como esto:**

30 LET x = x + 1 AND x < 31

La forma en que funciona esta línea es bastante compleja y se explica mejor en las partes del libro que tratan de instrucciones condicionales. Simplificadamente se puede decir que "si X es menor que 31 se añade 1 a su valor, pero si no es menor que 31, X se hace 0". Elimínese ahora la línea 40 (la que hemos hecho redundante) y renumérese en pasos de 10. Obtendremos el siguiente listado:

```
10 LET X=0
20 LET P=X
30 LET X=X+1 AND X<31
40 PRINT AT 5,P;" ";AT 5,X;"■"
50 GO TO 20
```

Se habrá advertido que en el espacio vacío se presenta una columna detrás del cuadrado todas las veces y, por consiguiente, pudiera ser posible expresar simplemente PRINT AT 5,X; "(espacio)■" y eliminar al propio tiempo la variable P. Esto hace la presentación más regular pero causa problemas cuando se llega al final de la línea y se necesita otra para resolverlos. Para ver lo que se quiere decir, pruébese este programa:

```
10 LET X=0
20 LET X=X+1 AND X<30
30 PRINT AT 5,X;" ■"
40 GO TO 20
```

¿Se aprecia lo que decíamos? Cada vez que se dispara un nuevo cuadro, el viejo permanece en el lado derecho de la pantalla. Permítasenos añadir un artificio para borrar estos cuadrados:

```
10 LET X=0
20 LET X=X+1 AND X<30
30 PRINT AT 5,X;" ■"
35 IF X=0 THEN PRINT AT 5,31;"
"
40 GO TO 20
```

Nuevamente podemos utilizar AND para acortar un poco el programa:

```
10 LET X=0
20 LET X=X+1 AND X<30
```

```
30 PRINT AT 5,X;" ■";AT 5,31;"
" AND X=0
40 GO TO 20
```

La línea 35 del primer programa y la segunda parte de la 30 en el segundo programa aseguran que sólo se produce el espacio vacío si el cuadrado ha alcanzado el final de su trayecto. He aquí otra forma de hacer esto mediante la variable de control de un bucle FOR/NEXT en lugar de la variable X convencional. Este método utiliza un poco menos de memoria y es algo más rápido.

```
10 FOR X=0 TO 30
20 PRINT AT 5,X;" ■";AT 5,31;"
" AND X=0
30 NEXT X
40 GO TO 10
```

Otra forma de usar este método es con un PRINT AT 5,31; "espacio)□" fuera del citado bucle FOR/NEXT. Tiene la ventaja de que el ordenador no ha de examinar la expresión condicional con tanta frecuencia por lo que el programa se ejecuta en forma mucho más rápida:

```
10 FOR X=0 TO 30
20 PRINT AT 5,X;" ■"
30 NEXT X
40 PRINT AT 5,31;" "
50 GO TO 10
```

Hemos acabado así con una rutina que es bastante rápida de ejecutar y económica en utilización de memoria. También hemos visto algunos de los problemas de la ejecución de esta clase de programas. Los ejemplos examinados hasta ahora tratan del movimiento constante a través de la pantalla. Será preciso también desplazar caracteres por toda la pantalla, posiblemente con el control de este movimiento por medio del teclado, de forma que el operador pueda realizar este control. Para hacer esto primeramente tenemos que regresar a la función INKEY$, una ayuda muy útil para el movimiento de gráficos. Si se ha leído una sección situada al principio de la obra, recordará que INKEY$ es el carácter que corresponde a la tecla que se pulsa en el teclado. Si se presiona la K, INKEY$ es "K", o bien, INKEY$ = "k". Si no se pulsa ninguna tecla, INKEY$ se convierte en una cadena vacía, o "". (Tal función no puede aplicarse a la tecla espaciadora porque

cuando se pulsa ésta actúa como ruptora de un Programa [BREAK] y detiene un programa de BASIC cuando está en ejecución.) Vamos a tratar ahora de algunas otras formas en que puede utilizarse la función INKEY$, ya que es muy poderosa y muy útil para el movimiento de gráficos. La mayoría de los juegos electrónicos comercializados requieren que se pulse un botón, se accionen conmutadores o que se manipulen ciertas palancas. El ZX Spectrum no dispone de tales elementos (a menos que los compre separadamente o se los construya) por lo que todo el control ha de hacerse con el teclado.

Para el control de movimiento en la pantalla ciertas teclas tienen ventajas sobre otras en virtud de su situación. Por ejemplo, la Z y la M en la hilera inferior del teclado. Pueden usarse para movimientos a izquierda (Z) y derecha (M). Su ventaja es que están lógicamente situadas para una utilización adecuada. Una ventaja es que la M está cerca de la tecla de ruptura (BREAK), por lo que accidentalmente, y ante la urgencia de los desplazamientos para evitar ser bombardeado por la "Gran Amenaza", pudiera detenerse el programa.

Si se necesita controlar desplazamientos a izquierda y derecha, arriba y abajo (como es el caso del movimiento del cursor en el programa de proceso de textos ["word processor"] al final del libro), la mejor elección son las teclas de los números 5, 6, 7 y 8, ya que incorporan unas flechas cuyas puntas señalan el movimiento. Están próximas entre sí para un uso adecuado y no se hallan peligrosamente cerca de BREAK.

La forma más común de utilizar INKEY$ en un programa de movimiento de gráficos (tanto en juegos como en aplicaciones más serias) es usarla en una instrucción condicional para controlar el valor de una variable. Por ejemplo:

```
IF INKEY$ = "8" THEN LET X = X + 1
```

Con ello se añade uno al valor de X, si se ha pulsado la tecla 8, pero X conserva su valor si no se acciona ninguna tecla, o se hace en una que no sea 8. Una vez se ha hecho esto, la variable puede utilizarse para controlar, por ejemplo, en qué parte de la pantalla hacer una presentación. Veamos un caso:

```
PRINT AT 5,X; "□"
```

Si adoptamos la convención de que X corresponda al eje horizontal e Y al vertical de la pantalla, cuanto mayor sea el valor de X

tanto más a la derecha se encontrará la representación, y cuanto mayor sea Y tanto más hacia la parte inferior de la pantalla se hallará la presentación. Sabiendo esto se puede redactar un pequeño programa para controlar el movimiento de un carácter (un asterisco, por ejemplo) en la pantalla mediante las teclas 5, 6, 7, 8 (las que tienen flecha):

```
 100>IF INKEY$="5" THEN LET X=X-
1
 110 IF INKEY$="8" THEN LET X=X+
1
 120 IF INKEY$="7" THEN LET Y=Y-
1
 130 IF INKEY$="6" THEN LET Y=Y+
1
 140 PRINT AT Y,X;"*"
```

Antes de poder ejecutar este programa, necesitamos definir X e Y o no ocurrirá nada. Añádanse estas líneas al programa:

```
10 LET X=0
20 LET Y=0
```

Ahora podemos ejecutar el programa. Lo que tenemos es un asterisco en la esquina superior izquierda de la pantalla y el programa se detiene después de la línea 140. Para impedir que esto suceda podemos añadir una línea como:

```
200>GO TO 100
```

Esto asegura que el ordenador continúa realizando su tarea una y otra vez. Ejecute el programa y pruebe a pulsar las teclas 5, 6, 7 y 8 de una en una. Se verá cómo el asterisco se mueve dejando detrás una estela de ellos. Continúe llegando hasta el límite de la pantalla y siga pulsando la tecla. Empiezan a ocurrir cosas extrañas. Si se alcanza el fondo de la pantalla o su extremo derecho, el programa se detendrá con un mensaje de error. Esto es debido a que el valor de X se ha hecho mayor que 31, o que el de Y ha superado 21 y, en consecuencia, se pide al ordenador que haga la presentación fuera de los límites de la pantalla lo que, por supuesto, no puede hacer.

Sin embargo, si se intenta que el movimiento traspase tales límites empezarán a suceder cosas más extrañas. ¡El asterisco empieza a desplazarse en dirección opuesta! No es que se haya estropeado el

ordenador, ni nada por el estilo, hay una simple explicación. Cuando el valor de X o el de Y es negativo (lo que sucede cuando tratamos de salirnos de la pantalla), la instrucción PRINT ignora el signo negativo (toma el valor absoluto [ABS], si se prefiere), por lo que aparentemente algunas teclas ¡cambian de función! No es muy útil, que digamos, esta explicación, ya que lo que necesitamos es que el asterisco regrese a la pantalla y se restaure el funcionamiento normal. Es preciso un método por el que el asterisco llegue al borde de la pantalla, se detenga y no intente salirse.

El método más sencillo es impedir que X e Y tomen valores que puedan causar estos problemas. Los valores admisibles para X son de 0 a 31, y para Y, de 0 a 21. He aquí cómo hacerlo. Cámbiense las líneas 100, 110, 120 y 130 para que queden como sigue:

```
 100>IF INKEY$="5" AND X>0 THEN
LET X=X-1
 110 IF INKEY$="8" AND X<31 THEN
 LET X=X+1
 120 IF INKEY$="7" AND Y>0 THEN
LET Y=Y-1
 130 IF INKEY$="6" AND Y<21 THEN
 LET Y=Y+1
```

Esto hace que el asterisco permanezca como es debido en la pantalla pero todavía nos queda el problema de la cola de ellos que se forman en su movimiento. Esto se debe a que no hemos previsto el borrado de la posición anterior cuando los asteriscos se mantenían en movimiento. La mejor forma de hacerlo es utilizar un segundo juego de variables para recordar la posición previa del carácter y que si es diferente de la nueva se produzca un espacio vacío en tal posición.

Para ello añádanse estas líneas:

```
  50>LET A=X
  60 LET B=Y
 135 IF A<>X OR B<>Y THEN PRINT
AT B,A;" "
 200 GO TO 50
```

El programa que se tiene ahora en el ordenador debe ser así:

```
10 LET X=0
20 LET Y=0
50 LET A=X
60 LET B=Y
```

```
 100 IF INKEY$="5" AND X>0 THEN
LET X=X-1
 110 IF INKEY$="8" AND X<51 THEN
 LET X=X+1
 120 IF INKEY$="7" AND Y>0 THEN
LET Y=Y-1
 130 IF INKEY$="6" AND Y<21 THEN
 LET Y=Y+1
 135 IF A<>X OR B<>Y THEN PRINT
AT B,A;" "
 140 PRINT AT Y,X;"*"
 200 GO TO 50
```

Ahora tenemos el diseño básico de un programa de movimiento de gráficos. Cuando se tenga necesidad de preparar un juego sobre esta rutina, posiblemente tendremos que alterar algunos detalles o cambiar el orden de las instrucciones pero los principios implicados serán similares. Como hicimos anteriormente podremos reducir algo la rutina, por ejemplo:

```
  10 LET X=0
  20 LET Y=0
  50 LET A=X
  60 LET B=Y
 100 LET X=X-(INKEY$="5" AND X>0
)+(INKEY$="8" AND X<31)
 110 LET Y=Y-(INKEY$="7" AND Y>0
)+(INKEY$="6" AND Y<21)
 135 IF A<>X OR B<>Y THEN PRINT
AT B,A;" "
 140 PRINT AT Y,X;"*"
 200 GO TO 50
```

Esta versión ocupa casi cincuenta "bytes" de memoria menos que la anterior versión. Puede combinarse la línea 140 con la 135 para formar una instrucción condicional ya que no se precisa de ninguna de las instrucciones PRINT a menos que se mueva el asterisco. Esto significa que podemos eliminar la línea 140. Veamos cómo cambiar la línea 135:

```
 135>IF A<>X OR B<>Y THEN PRINT
AT B,A;" ";AT Y,X;"*"
```

Recuérdese eliminar la línea 140. Existe el inconveniente de que si bien se ahorran cinco "bytes" más de memoria, cuando se ejecuta la primera vez el programa no aparece realmente el asterisco sobre la

pantalla hasta que se mueve. Por lo que quizá sea mejor no hacer esto a no ser que se tenga una necesidad extrema de capacidad de memoria.

SCREEN$ y desplazamiento en pantalla

Veamos ahora otro artificio útil para los programas de movimiento de gráficos: el desplazamiento ("scrolling"). Este movimiento puede hacerse automáticamente incluyendo la instrucción POKE 23692, –1, que desplaza toda la presentación una línea más arriba, perdiéndose todo lo que haya en el renglón superior.

El programa siguiente —CARRERAS DE COCHES—, muestra nuevamente la acción del desplazamiento para producir gráficos en movimiento. En este programa se intenta conducir una línea larga de pequeños cochecitos descendiendo por una pista zigzagueante de asteriscos rojos. Los controles son las teclas "Z" y "M" que mueven a izquierda y derecha respectivamente.

Las líneas 80 y 90 mueven la pista aleatoriamente, asegurando que no se sale de los límites de la pantalla. La línea 110 presenta el coche que es desplazado (como ocurre con la pista) por las líneas 130 y 140.

La línea 160 introduce una nueva función del Spectrum, SCREEN$, que retorna información sobre el estado de la pantalla en el sitio que se especifica de acuerdo con la palabra SCREEN$. En la línea 160, el ordenador utiliza SCREEN$ para mirar en la celda del carácter justamente en frente de donde se presentó el último coche. Si encuentra un asterisco allí, sabe que el automóvil está a punto de chocar, enviando la acción a la línea 200 donde empieza la rutina "SE HA ESTRELLADO".

```
10 REM CARRERAS DE COCHES
20 LET C=0
30 LET T=0
40 GO SUB 250
50 LET A=10
60 LET X=13
```

```
  70 LET Y=12
  80 LET K=INT (RND*2)
  90 LET A=A-(K=1 AND A>1)+(K=0
AND A<24)
 100 REM  LA SIGUIENTE LINEA CON
TIENE UN GRAFICO C COMO LA 200
 110 PRINT AT Y,X-1; INK 1;"#"
 120 PRINT AT 20,A; INK 2;"*";TA
B A+5;"*"
 130 PRINT
 140 POKE 23692,-1: PRINT
 150 PRINT INK 6; PAPER 2;AT 0,1
0;" PUNTOS ";T;"  "
 160 IF SCREEN$ (Y+1,X-1)="*" TH
EN GO TO 200
 170 LET X=X-(INKEY$="Z")+(INKEY
$="M")
 180 LET T=T+1
 190 GO TO 80
 200 PRINT AT Y,X-1; INK 1;"#"
 210 PRINT AT 6,8; FLASH 1; BRIG
HT 1;" TE HAS ESTRELLADO!!  "
 220 PRINT AT 8,10; FLASH 1; BRI
GHT 1; INK RND*7; PAPER 9;" TU P
UNTUACION ES ";T;"  "
 230 BEEP .01,RND*20-RND*20
 240 GO TO 210
 250 FOR J=0 TO 7
 260 READ Z
 270 POKE USR "C"+J,Z
 280 NEXT J
 290 RETURN
 300 DATA BIN 00110110,BIN 00110
110,BIN 00111110,BIN 00010100,B
IN 00111110,BIN 00110110,BIN 00
011100,0
```

A continuación siguen otros varios programas para ilustrar las ideas sobre gráficos en movimiento de las que hemos tratado.

La flecha roja

El objeto de este juego es disparar, pulsando cualquier tecla excepto BREAK, a la FLECHA ROJA cuando vuela sobre su base y si hay acierto se apunta un tanto. Sólo se dispone de un número limitado de disparos, por lo que hay que tratar de alcanzar la mayor puntuación posible antes de que finalice el juego.

He aquí el listado:

```
   5 REM LA LINEA 10 CONTIENE
   7 REM LOS GRAFICOS A,B,C
   8 GO SUB 200
   9 PAPER 7: CLS
  10 PRINT AT 15,14; INK 1;"/—\"
;AT 4,0; PAPER 7; INK 2;"█████IM
PACTOS:0█████";AT 16,0; INK 3;"█
████████████████████
  20 LET H=0
  30 LET M=H
  40 LET Y=RND*6+7
  50 FOR X=0 TO 19
  60 IF H+M=250 THEN GO TO 1000
  70 REM EL GRAFICO D EN LA 80
  80 PRINT AT Y,X; INK 2;" »"
  90 IF INKEY$<>"" AND X=14 THEN
 GO TO 140
 100 IF INKEY$<>"" THEN LET M=M+
1
 110 NEXT X
 120 PRINT AT Y,20;" "
 130 GO TO 40
 140 LET H=H+1
 150 FOR F=1 TO 30
 160 PRINT AT Y,X-2; INK 2; PAPE
R 4; BRIGHT 1; FLASH 1;"IMPACTO"
; FOR Z=1 TO 20: BORDER RND*7: B
EEP .01,39-3*Z: PRINT AT Y,X-2;"
       "
 170 NEXT F
 180 PRINT AT 4,14; PAPER 7; INK
 2;H
 190 GO TO 40
 200 FOR J=0 TO 7
 210 READ Q
 220 POKE USR "/"+J,Q
 230 NEXT J
 240 FOR J=0 TO 7
 250 READ Q
 260 POKE USR "—"+J,Q
 270 NEXT J
 280 FOR J=0 TO 7
 290 READ Q
 300 POKE USR "\"+J,Q
 310 NEXT J
 320 FOR J=0 TO 7
 330 READ Q
 340 POKE USR "»"+J,Q
 350 NEXT J
 400 RETURN
 500 DATA BIN 00000001,BIN 00000
011,BIN 00000110,BIN 00001100,BI
N 00011000,BIN 00110000,BIN 0110
000,BIN 11000000
 510 DATA BIN 11111111,BIN 11111
111,0,0,0,0,0,0
 520 DATA BIN 11100000,BIN 11100
000,BIN 00110000,BIN 00011000,BI
```

```
N 00001100,BIN 00000110,BIN 0000
0011,1
 530 DATA BIN 00001000,BIN 10000
100,BIN 01000010,BIN 00111111,BI
N 01000010,BIN 10000100,BIN 0000
1000,0
1000 FOR G=1 TO 70
1010 BEEP .01,G/2: BEEP .01,40-G
1020 NEXT G
1030 RUN
```

La línea 10 establece la parte principal de la presentac ón, la que no se mueve durante el programa. H es la variable que recuerda la cantidad de impactos que puntúan y M la de fallos (disparos que no alcanzan al OVNI). Ambas están inicialmente puestas a cero. La línea 40 establece la ordenada Y de la FLECHA en un valor aleatorio del 7 al 13. La instrucción PRINT toma el valor del número entero más próximo si una de sus coordenadas no es un entero por lo que RND*6 + 7 varía de 7 a 13, como decíamos. Esto genera la posición de la FLECHA en la parte superior de la pantalla. El bucle principal que controla el vuelo de la nave en sentido transversal de la pantalla empieza en la línea 50. La posición de partida es 0, y la final, 19. La línea 60 comprueba si se han hecho todos los disparos y, en caso afirmativo, remite el programa a la línea 1000. La instrucción 80 presenta la FLECHA. Obsérvese cómo se borra la posición anterior a la que ocupa en cada instante el móvil mediante el espacio vacío que hay detrás de él. La línea 90 comprueba si se ha pulsado una tecla y si la FLECHA se halla justamente encima de la base, enviando después la acción a la rutina IMPACTO en la línea 140. Esta añade 1 a la puntuación y produce una presentación de una llamativa explosión en el punto del impacto, remitiendo después el programa a la línea 40 nuevamente para producir otra FLECHA. Sin embargo, si se ha pulsado una tecla y la nave no estaba encima de la base, se añade 1 al número de fallos. Cuando la FLECHA alcanza el final de su trayectoria, el programa sale del ciclo FOR/NEXT y la línea 120 borra la posición final del aparato volador, mientras que la 130 retorna el programa a la línea 40 para empezar de nuevo.

El engullidor de basura

En este juego se trata de eliminar la basura que aparece en la pantalla en forma de configuraciones color magenta. Se controla el "engullidor" con las teclas 5, 6, 7 y 8 con los movimientos correspondientes a las direcciones de las flechas que tienen dichas teclas. La basura es engullida al pasar sobre ella; se dispone de un tiempo limitado y se expresa en cada momento los elementos de basura que se han recogido. El bucle de las líneas 40 a 60 establece aleatoriamente las posiciones de la basura.

```
  10 REM ENGULLIDOR DE BASURA
  20 GO SUB 1000
  30 REM GRAFICO M EN 50
  40 FOR G=1 TO 50
  50 PRINT AT RND*16+3,RND*26+4;
 INK 1;"▒"
  60 NEXT G
  70 LET X=0
  80 LET Y=0
  85 FOR G=1 TO 500
  90 LET A=X
 100 LET B=Y
 110 LET Y=Y-(INKEY$="7" AND Y>1
)+(INKEY$="6" AND Y<20)
 120 LET X=X-(INKEY$="5" AND X>1
)+(INKEY$="8" AND X<31)
 130 IF ATTR (Y,X)=49 THEN LET P
UNTOS=PUNTOS+1: PRINT AT 0,8; IN
K RND*7; PAPER 9; FLASH 1;"LA PU
NTUACION ES ";PUNTOS;AT 20,0;"TI
EMPO RESTANTE = ";500-G;"  "
 140 IF ATTR (Y,X)=49 THEN BEEP
.01,40
 150 REM GRAFICO D EN 160
 160 PRINT AT B,A;" ";AT Y,X; IN
K 2;"◀"
 170 NEXT G
 180 PRINT AT 0,8; INK RND*7; PA
PER 9; FLASH 1;"LA PUNTUACION ES
 ";PUNTOS;AT 20,0;"TIEMPO RESTAN
TE = ";0;"  "
 190 BEEP .1,RND*50: GO TO 190
 900 STOP
1000 FOR A=0 TO 7
1010 READ Q
1020 POKE USR "▒"+A,Q
1030 NEXT A
1040 DATA BIN 10100101,BIN 01011
010,BIN 10100101,BIN 01011010,BI
N 01011010,BIN 10100101,BIN 0101
```

```
1010,BIN 10100101
1050 FOR A=0 TO 7
1060 READ Q
1070 POKE USR "C"+A,Q
1080 NEXT A
1090 DATA BIN 00011111,BIN 01111
111,BIN 11111100,BIN 11111000,BI
N 11100000,BIN 11111000,BIN 0111
1100,BIN 00111111
1100 BORDER 2: PAPER 6: CLS
1120 LET PUNTOS=0
1500 RETURN
```

Ahora podemos explicar cómo saber lo que ocurre en la pantalla de forma que el ordenador se entere de que el "engullidor" va a pasar sobre algo. Este artificio se utilizará multitud de veces en los programas. La función esencial es ATTR que, como se ve en las líneas 130 y 140, es similar en forma al enunciado PRINT AT. Las líneas 130 y 140 comprueban los atributos del carácter del cuadrado Y,X antes de que se represente allí el "engullidor". Si se encuentra un valor de 49, sabe que tal cuadrado contiene basura, con lo que el tiempo y la puntuación son actualizados. El número realmente producido por la función ATTR depende de que el carácter destellee (FLASHING) o no, de que se le aplique o no brillo (BRIGHT) y de los colores de la tinta (INK) y del fondo (PAPER) en la posición considerada.

Debido a que la función ATTR es un poco difícil de utilizar, es mejor establecer una pequeña rutina para que presente los resultados de dicha instrucción antes de que se decida qué valor se desea probar. La "ausencia" de un valor producido cuando el "engullidor" pasa sobre un espacio vacío no es recomendable; en lugar de esto, compruébese la presencia de uno determinado. Así lo hicimos al escribir el programa presente, al tener la línea 140 vacía al principio, y la 130 con PRINT AT 0,0;ATTR(Y,X), y observando después lo que sucedía cuando el "engullidor" iba a pasar sobre un cuadrado con basura. Recomendamos que se siga este método.

La manipulación de cadenas alfanuméricas

Finalmente, hay otro método para producir gráficos en movimiento en BASIC, por medio de variables en cadena o alfanuméricas que

no se utiliza muy frecuentemente. Estas cadenas pueden representarse muy rápidamente, y las grandes posibilidades del ordenador para su fraccionamiento (tratado anteriormente) nos permiten disponer de una poderosa herramienta. El método básico consiste en asignar un caracter o conjunto en cadena suficientemente largo para cubrir la zona de la pantalla utilizada para su movimiento y presentación en una localización determinada. Para simular el movimiento se pueden presentar en pantalla diferentes partes de conjunto o cambiar el contenido de dicho conjunto.

Preséntense en pantalla diferentes partes de la cadena por orden. Pruebe este corto programa:

```
 10 DIM A$(32)
 20 LET A$(1)="+"
 30 FOR A=32 TO 1 STEP -1
 40 PRINT AT 20,0;A$(A TO )+A$(
TO A-1)
 50 NEXT A
 60 GO TO 30
```

¿Puede verse por qué es necesario que la línea 30 cuente *al revés* de 32 a 1? ¿Qué sucedería si dicha línea contase en sentido directo del 1 al 32 (30 FOR A = 1 TO 32)? Dibújese en un trozo de papel (o utilice la impresora, si se dispone) cada una de las fases del programa necesarias para hacer la presentación. Adviértase la gran velocidad posible y cómo se hace simultáneamente la presentación en una posición y el borrado de la precedente. Es posible conseguir un efecto interesante cambiando la línea 20 para que diga 20 INPUT A$ e introdúzcase un mensaje de hasta 32 caracteres. Esto es similar a un tipo de presentación que se hace en los escaparates con fines publicitarios aunque no se trata de la clase de efecto que uno quisiera encontrar normalmente.

Este método de producir gráficos en movimiento a partir de cadenas es muy útil porque no altera el contenido de las cadenas/conjuntos, sino que las presenta en orden diferente por lo que su información se puede recuperar fácilmente en cualquier momento. Pueden servir para una presentación de escaparate como la que sigue a continuación.

```
20 INPUT "ENTRE SU MENSAJE";A$
30 IF A$="" THEN GO TO 20
40 LET S=50
50 FOR B=1 TO LEN A$+33
```

```
  70 PRINT AT 5,0; ("
                         "+A$+"
                              ") (B T
O B+31)
  80 LET S=S-(5 AND INKEY$="F" A
ND S>1)+(5 AND INKEY$="D")
  90 IF INKEY$="A" THEN GO SUB 1
40
 100 FOR A=1 TO S
 110 NEXT A
 120 NEXT B
 130 GO TO 50
 140 FOR A=1 TO 200
 150 NEXT A
 160 RETURN
```

Lo que hace es pedir que se introduzca un mensaje de presentación; cuando se ha hecho, tal mensaje empieza a aparecer desde la derecha de la pantalla hasta la izquierda por donde desaparece, repitiéndose nuevamente la secuencia. La presentación empieza a una velocidad lenta pero puede acelerarse presionando la tecla "F" y hacerse menos rápida con la tecla "D". La velocidad más alta lo es mucho y la lenta es muy reducida. Se puede "congelar" la presentación por un corto espacio de tiempo pulsando la tecla "A". En cualquier momento, es posible detener el programa mediante el mando BREAK.

Si el ordenador tiene suficiente memoria disponible, el tamaño del mensaje se limita teóricamente por el del mayor conjunto que pueda manejar. En la práctica, sin embargo, si se llena la pantalla con el mensaje cuando se entra (es decir, tiene más de 24 líneas de 32 caracteres cada una), los que faltan han de introducirse a ciegas pues no caben en la pantalla. Trate de hacerlo. ¡Acabará con los dedos cansados!

El mensaje empieza a aparecer desde el lado derecho de la pantalla y se desplaza en ella hacia la izquierda. Una vez que ha desaparecido por dicho lado reaparece como antes y el ciclo se repite una y otra vez. La velocidad puede variarse como se ha dicho antes y lo mismo sucede con la "congelación" de la representación en pantalla.

La línea 70 es bastante compleja: para impedir cambiar el contenido de la cadena A$, todo lo que se halla dentro del primer par de paréntesis es tratado como una larga cadena compuesta de treinta y dos espacios, seguida de la cadena mensaje A$ y, a continuación, otros treinta y dos espacios. El fraccionador de cadenas en el segundo par de paréntesis selecciona qué partes de esta larga cadena han de ser presentadas. Obsérvese que A$ todavía conserva su propia identidad. Cualesquiera que sean las partes que se seleccionen, la cadena presentada tiene siempre una longitud de 32 caracteres.

Tal como está, el programa no permite cambiar el mensaje una vez en ejecución; hay que detenerlo con BREAK y ejecutarlo nuevamente. Una forma de poder hacer el cambio es añadiendo esta línea, de manera que al pulsar "1"(EDIT) el programa vuelva a iniciarse automáticamente.:

```
85 IF INKEY$="1" THEN RUN
```

Movimiento de los elementos del conjunto (array). Pruébese este programa:

```
10 DIM A$(32)
20 FOR A=1 TO 32
30 LET A$(A)="+"
40 PRINT AT 20,0;A$
50 LET A$(A)=" "
60 NEXT A
70 GO TO 20
```

Este método nos proporciona una gran flexibilidad. Pueden manejarse cadenas rápida y eficientemente haciendo uso de las posibilidades del ordenador para operar con ellas. Las cadenas son muy útiles para almacenar información ya que se puede tener acceso a ella con rapidez y facilidad en comparación, sobre todo, con las instrucciones REM; la velocidad con que pueden ser presentadas en pantalla las convierte en un método atractivo para las presentaciones. La principal desventaja es que se pierde mucha memoria ya que la información se mantiene tanto en el archivo de presentación como en los conjuntos (arrays) implicados, y posiblemente también en el área del programa. He aquí un programa de presentación de movimiento que se basa en la información que hay en las cadenas presentadas.

El programa se denomina "BASIC Invaders" porque es una versión del juego comercializado "LOS INVASORES" escrito en BASIC. Necesariamente está muy simplificado y se incluye con el fin de demostrar el uso de las cadenas para presentaciones en movimiento. Una ola de invasores desciende amenazadora hacia usted que se puede mover a izquierda y derecha con las teclas 5 y 8 respectivamente. Se dispara a los invasores con la tecla 7. Si usted está directamente bajo el invasor al disparar, la nave queda destruida y desaparece.

Hay siete olas de invasores y se precisa destruirlos a todos para ganar.

```
   1 REM LOS INVASORES
   2 REM GRAFICO DE F EN 40
   3 REM GRAFICO DE B EN 120
   4 BORDER 2: PAPER 0: CLS
   5 GO SUB 290
  10 DIM A$(32)
  20 DIM B$(32)
  30 FOR D=1 TO 7
  40 LET A$="      Ѫ    Ѫ    Ѫ    Ѫ
 Ѫ    Ѫ   "
  50 LET X=INT (RND*32)
  60 LET C=X
  70 PRINT AT 5,12; INK X/5; PAP
ER 9; FLASH 1; BRIGHT 1;" OLA :
";D;" "
  80 LET C=0
  90 FOR B=D+9 TO 19 STEP 2
 100 FOR A=0 TO 31
 110 LET X=X+(INKEY$="8" AND X<3
1)-(INKEY$="5" AND X>0)
 120 PRINT AT B,0; INK RND*2;A$;
AT 20,C;" " AND C<>X;AT 20,X; IN
K 4;"Ѧ"
 130 IF A$=B$ THEN GO TO 200+(60
 AND D=7)
 140 LET C=X
 150 IF INKEY$="7" THEN BEEP .01
,20: IF A$(X+1)="Ѫ" THEN LET A$(
X+1)="█": PRINT AT B,0; INK 6;A$
: BEEP .01,50: LET A$(X+1)=" ":
LET SC=SC+1: PRINT AT 2,7; INK 2
; PAPER 0; BRIGHT 1; FLASH 1;"LA
 PUNTUACION ES ";SC*27187
 160 NEXT A
 165 LET A$=A$(4 TO )+A$( TO 3)
 170 PRINT AT B,0;B$
 180 NEXT B
 190 GO TO 240
 200 FOR B=1 TO 10
 210 NEXT B
 220 PRINT AT 20,C;" "
 230 NEXT D
 240 PRINT FLASH 1; INK RND*7; P
APER 9;"HAN ATERRIZADO!!"
 250 BEEP .1,-RND*30: POKE 23692
,-1: GO TO 240
 260 PRINT INK RND*7; PAPER 9;"T
ODOS LOS INVASORES DESTRUIDOS"
 265 POKE 23692,-1
 270 BEEP .01,RND*50: GO TO 260
 280 REM COMPOSICION DE LOS INVA
SORES
 290 FOR J=0 TO 7
 300 READ Q
```

```
 310 POKE USR "■"+J,Q
 320 NEXT J
 330 DATA BIN 00111110,BIN 00101
010,BIN 00111110,BIN 00011100,BI
N 00001000,BIN 01110111,BIN 0100
0001,BIN 01000001
 340 REM COMPOSICION DE LA BASE
 350 FOR J=0 TO 7
 360 READ Q
 370 POKE USR "■"+J,Q
 380 NEXT J
 390 DATA BIN 01111110,BIN 01111
110,BIN 01111110,BIN 01111110,BI
N 01100110,BIN 11100011,BIN 1100
0011,BIN 10000001,BIN 10000001
 400 LET SC=0
 410 RETURN
```

El programa derrocha mucha memoria. No se ha intentado conservarla y es posible que usted pueda ganar velocidad con modificaciones menores. Las dos cadenas interesantes son A$ y B$. La segunda establece 32 espacios y se utiliza para evitar que haya que pulsar "(32 espacios)" en varios puntos del programa. A$ es la cadena que representa a los invasores, y se establece inicialmente en 32 elementos, el número de caracteres es una línea de la presentación. La línea 40 pone el estado inicial de los caracteres y puede ser cualquier combinación de espacios y gráfico y podría componerse precisamente de 32 caracteres. La variable que controla la posición del jugador es X y su valor se altera en la línea 110. La 120 realiza la presentación principa, actualizando la de los invasores y la de la posición del jugador.

La línea 130 compara los invasores con una cadena de 32 espacios (B$) y si de la comparación se deduce que A$ no contiene invasores (es decir, sólo se trata de espacios vacíos) salta a la siguiente ola de atacantes o, si se está en la última ola, pasa al mensaje de victoria de la línea 260. La 150 tiene particular interés ya que es la que analiza la cadena en busca del carácter que está justamente encima del defensor y si encuentra un invasor allí lo convierte en un espacio vacío. Esto sólo se realiza si está pulsada la tecla 7.

El resto del programa se refiere principalmente al tratamiento de tiempo de los bucles y de la disposición de las diferentes olas de invasores.

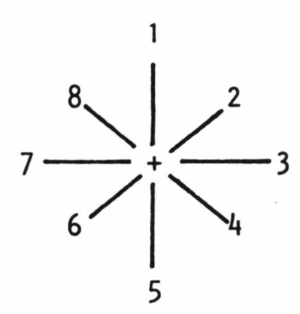
Si se está conservando la totalidad de la información de pantalla en una cadena-conjunto (o una parte de aquélla implicando líneas por encima o por debajo), pueden utilizarse dos métodos diferentes empleando distintos tipos de conjuntos. Considérese el caso de toda la pantalla de 22 por 32. Se requerirá una cadena/conjunto (array) de 22 por 32 elementos y esto se puede lograr mediante:

(1) Un conjunto (array) de dos dimensiones, establecido por la instrucción DIM A$(22, 32). Se puede utilizar la instrucción PRINT AT en las coordenadas para tener acceso a los elementos, recordando que los elementos del conjunto (array) empiezan con 1 y las coordenadas de PRINT con 0. Por ejemplo, para una presentación en Y,X del carácter T (PRINT AT Y,X;CHR$ T) se escribiría LET A$ (Y + 1,X + 1) = CHR$ T. El problema que tenemos con este método es que se precisa una larga introducción PRINT AT 0,0;A$(1);A$(3) ... A$(21), A$(22). Sin embargo, puesto que la razón principal de utilizar un conjunto (array) para la presentación es la facilidad de acceso a la información, esto sólo es necesario al principio de un programa ya que a partir de entonces solamente es preciso presentar (PRINT) las partes del conjunto (array) con las que realmente estamos tratando. Tomemos, como ejemplo, el juego de las damas. Necesitaríamos todo el tablero en la pantalla al principio del juego pues precisamos mirarlo en su totalidad; sin embargo, cuando llegara el momento de presentar movimientos sólo sería necesario presentar las partes del conjunto (array) que cambian con el movimiento, la parte a la que se desplaza la pieza y posiblemente otra donde se capturara una pieza adversaria.

(2) Utilícese un conjunto (array) unidimensional con 704 elementos para una presentación de 22 por 32, estableciendo la sentencia DIM A$ (704). Esto puede tratarse como un mapa de memoria en pantalla y es posible presentarla de una vez con la instrucción PRINT AT 0,0;A$, que es muy rápida de ejecución. Los elementos tienen fácil acceso. Para mover un carácter sobre la pantalla tenemos que hacerlo en el conjunto (array) de forma que se desplace satisfactoriamente. Para comprender cómo hacer esto hay que saber la disposición del conjunto (array) en la pantalla. He aquí un diagrama:

X Y	Ø	1	2	29	3Ø	31
Ø	A$(1)	A$(2)	A$(3)	A$(3Ø)	A$(31)	A$(32)
1	A$(33)	A$(34)	A$(35)	A$(62)	A$(63)	A$(64)
2	A$(65)	A$(66)	A$(67)	A$(94)	A$(95)	A$(96)
3	A$(97)	A$(98)	A$(99)	A$(126)	A$(127)	A$(128)

La figura muestra un fragmento de la parte superior de la pantalla. Y y X son las coordenadas de PRINT AT Y,X;función. ¿Puede verse cómo se relacionan Y,X con los subíndices de A$? Hay 32 elementos de A$ en cada línea de la presentación. La X empieza en 0 mientras que los subíndices del conjunto (array) empiezan en 1 por lo que Y,X se corresponden con A$(Y*32 + X + 1). Cuando se mueve un carácter puede ir a una de las ocho localizaciones que le rodean, como se ve en la figura que sigue:



Supongamos ahora que el invasor está en A$(A). Sigue una tabla que muestra las diferencias entre los subíndices de los elementos que representan las posibles posiciones por todos los lados (cuanto hay que añadir al viejo subíndice para hacer el nuevo).

Dirección del movimiento	*Cuánto hay que añadir a A*
1	−32
2	−31
3	1
4	33
5	32
6	31
7	−1
8	−33

INKEY$ cambiado

En este momento hemos de tener cuidado de no pasarnos los límites del conjunto (array), lo que sucedería con la utilización de la instrucción PRINT para evitar que el programa se destruya con un subíndice de error. Se pueden utilizar las teclas del cursor para controlar los movimientos verticales y horizontales, y las del cursor cambiado para los diagonales. A continuación sigue un corto programa que mueve un gráfico por toda la pantalla mediante control de cursor para ilustrar cómo puede hacerse.

Pulsando las teclas 5, 6, 7 y 8 simultáneamente con una tecla SHIFT se efectúa el movimiento con un ángulo de 45° en sentido directo con respecto a la dirección de las flechas que hay en las cuatro teclas primeras. Estas solas producen movimientos en la dirección de las flechas. Este programa no permite impedir que ocurran errores de los subíndices debido a movimientos fuera de los límites del conjunto.

```
  10 DIM A$(704)
  20 LET A=INT (RND*704)+1
  30 LET A$(A)=" "
  40 LET A=A-(32 AND INKEY$="7")
-(31 AND INKEY$=CHR$ 11)+(1 AND
INKEY$="8")+(33 AND INKEY$=CHR$
9)+(32 AND INKEY$="6")+(31 AND I
NKEY$=CHR$ 10)-(1 AND INKEY$="5"
)-(33 AND INKEY$=CHR$ 8)
  50 LET A$(A)="█"
  60 PRINT AT 0,0;A$
  70 GO TO 30
```

La razón por la que SHIFT-5, SHIFT-6, SHIFT-7 y SHIFT-8 se han representado por CHR$ 8, CHR$ 10, CHR$ 11 y CHR$ 9 es porque no pueden entrarse directamente desde el teclado (actúan como control del cursor, si se prueba), por lo que la forma más fácil de introducirlas es por medio de la función CHR$.

Debe destacarse que la utilización de las cadenas para crear gráficos en movimiento está limitada a aquellas aplicaciones donde la velocidad no tiene gran importancia mejor que en las que, aun siendo importante, no es excesivamente esencial, prefiriéndose la facilidad de acceso a la información. Un ejemplo lo tenemos en los juegos de tablero, como las damas, en los que las piezas se mueven ocasionalmente pero se precisa un rápido acceso a la información.

Introducción a la aritmética en el ordenador

Eche una rápida ojeada a esta sección antes de leerla en detalle ya que pudiera ser que no haya nueva información para usted. Si es este el caso, no se preocupe y pase las páginas hasta la próxima sección.

Los símbolos para las diversas operacioes en el lenguaje BASIC son probablemente bien conocidos a estas alturas. Son: multiplicación (*), división (/), substracción (—), adición (+) y potenciación (↑ — SHIFT H). El ordenador sigue un estricto orden de prioridad para sus operaciones.

Por prioridad se entiende el orden en que se evalúan las expresiones. El ordenador no sigue el de la presentación en pantalla. Por ejemplo, se habrá advertido que una expresión entre paréntesis podría producir un resultado diferente comparado al que se obtendría si se omitieran estos paréntesis (en realidad esta omisión puede dar lugar al rechazo del ordenador por error de sintaxis). Cada tipo de operación recibe un orden de *prioridad* que se mide con números del 1 al 16.

Las operaciones con el orden de prioridad más elevado son evaluadas primero y las que tienen el mismo se realizan de izquierda a derecha. En efecto, el ordenador mira a la expresión y observa la parte que tiene mayor prioridad y se dice a sí mismo: «Un momento, amigos, hay algo con preferencia allí. Volveré con vosotros en un minuto cuando llegue vuestro turno.''

Operación	*Prioridad*
()	16
Subindicado y fraccionamiento	12
Todas las cadenas	11
↑	10
— (negación)	9
*	8
/	8
+	6
— (substracción)	6
=,>.<,<=,>=,<'	5
NO (NOT)	4
Y (AND)	3
O (OR)	2

Téngase en cuenta que un número se supone positivo a menos que se halle precedido del signo menos. En forma similar, a no ser que aparezca el punto decimal (la coma en nuestras matemáticas) con un número, el ordenador lo considera entero. Aunque se puede utilizar el punto decimal en el ordenador, las comas no se admiten para facilitar la lectura de los números. El uso de la notación científica en números muy grandes y muy pequeños se explicó en la sección dedicada a las variables. Refiérase a ella si se precisa un recordatorio sobre dicha notación.

El BASIC en este ordenador funciona con bastante rapidez, como puede apreciarse con los siguientes programas. El primero de ellos trata de progresiones aritméticas. Se introduce el primer término, el número de términos y la razón; el ordenador producirá la información con suma rapidez.

```
  10 REM PROGRESION ARITMETICA
  20 PRINT "RESOLVERE PARA USTED
ES LA"
  30 PRINT "PROGRESION ARITMETIC
A A PARTIR"
  40 PRINT "DE LA INFORMACION DA
DA."
  50 PRINT '"DEME EL PRIMER TERM
INO"
  60 INPUT PRIMERO
  70 PRINT ,PRIMERO
  80 PRINT '"Y LA RAZON?"
  90 INPUT RAZON
 100 PRINT ,RAZON
 110 PRINT '"CUANTOS TERMINOS?"
 120 INPUT TERMINOS
 130 LET TERMINOS= INT (TERMINOS
+.5)
 140 CLS
 150 POKE 23692,-1
 160 PRINT INK 6; PAPER 0;"PROGR
ESION ARITMETICA"
 180 PRINT INK 2;'"TERMINO";TAB
13;"VALOR"
 190 LET SUMA=0
 200 FOR L=0 TO TERMINOS-1
 210 LET W=L+1
 220 LET Q=PRIMERO+(L*RAZON)
 230 LET SUMA=SUMA+Q
 240 PRINT TAB 4;W;TAB 13;Q
 250 NEXT L
 260 PRINT INK 2;TAB 4;'"LA SUMA
 ES ";SUMA
```

```
RESOLVERE PARA USTEDES LA
PROGRESION ARITMETICA A PARTIR
DE LA INFORMACION DADA.
```

```
DEME EL PRIMER TERMINO
                    234
Y LA RAZON?
                    13.5
CUANTOS TERMINOS?


PROGRESION ARITMETICA

TERMINO         VALOR
    1           234
    2           247.5
    3           261
    4           274.5
    5           288
    6           301.5
    7           315
    8           328.5
    9           342
    10          355.5
    11          369
    12          382.5

LA SUMA ES 3699
```

Como puede verse, el programa también determina la suma de los términos.

Números primos

Los números primos son fáciles de determinar.

```
  10 REM NUMEROS PRIMOS
  20 PRINT "ENTRAR EL VALOR DEL"
  30 PRINT "NUMERO PRIMO MAXIMO"
  40 PRINT TAB 4;"QUE SE DESEE"
  50 INPUT A
  60 DIM Z(A): LET KL=A
  70 FOR J=1 TO A: LET Z(J)=J
  80 NEXT J: IF A<4 THEN GO TO 2
00
  90 LET Z(4)=5: LET KL=4
 100 LET IZ=5
 110 LET IZ=IZ+2
```

```
 120 IF IZ>A THEN GO TO 200
 125 LET JO=3
 130 LET EX=IZ/Z(JO+1)
 140 IF EX=INT (EX) THEN GO TO 1
10
 150 IF EX<Z(JO+1) THEN GO TO 18
0
 160 LET JO=JO+1
 170 GO TO 130
 180 LET KL=KL+1: LET Z(KL)=IZ
 190 GO TO 110
 200 POKE 23692,-1
 210 PRINT "LOS NUMEROS PRIMOS H
ASTA EL ";A
 230 PRINT INK 2; PAPER 6;"NUM.D
E PRIMOS","PRIMOS"
 240 FOR C=1 TO KL
 250 PRINT TAB 4;C,Z(C)
 260 NEXT C
```

```
ENTRAR EL VALOR DEL
NUMERO PRIMO MAXIMO
    QUE SE DESEE
LOS NUMEROS PRIMOS HASTA EL 20
NUM.DE PRIMOS   PRIMOS
    1              1
    2              2
    3              3
    4              5
    5              7
    6              9
    7              11
    8              13
    9              17
    10             19
```

La capacidad matemática del ordenador puede utilizarse, por supuesto, para producir otra clase de información.

Un programa de estadística

El siguiente programa, una serie de rutinas estadísticas, puede fácilmente dividirse en otros cuatro más cortos aplicando los números de línea desde el 1000 al 1500, del 2000 al 2510, del 3000 al 3510 o del 4000 al 4100. También habrá que asignar el "COUNT" (LA CUENTA) y el TOTAL para cada programa.

Los cuatro programas son:

Media aritmética Es el promedio de un conjunto de números.

Media geométrica Es la raíz enésima del producto de los números, donde n es el total de ellos.

Media armónica La media armónica se deriva de los recíprocos de los números entrados.

Factorial Un factorial es el producto A*(A—1)*(A—2)*(A—3) ... *(2)*(1), donde A es el entero aplicado en la línea 4030. Como esta función sólo es posible con números enteros, la citada línea cambia cualquier número no entero en un entero.

La rutina desde la línea 9000 presenta un repertorio de elecciones. Adviértase el uso de GO TO A*1000 en la línea 9600. Esto es una forma abreviada de decir:

```
IF A = 1 THEN GO TO 1000
IF A = 2 THEN GO TO 2000
IF A = 3 THEN GO TO 3000
IF A = 4 THEN GO TO 4000
```

Se puede hacer frecuentemente uso de esta técnica en los programas mandados por un repertorio.

```
  20 GO TO 9000
  30 REM   EL ENTRECOMILLADO DEN
TRO DE LAS LINEAS 1040,2040,3040
 PROCEDE DE LA TECLA A
 900 REM *************************
1000 REM MEDIA ARITMETICA
1010 PRINT "MEDIA ARITMETICA"
1020 PRINT "INTRODUZCA LOS NUMER
OS"
1030 PRINT "QUE DESEE PROMEDIAR"
1040 PRINT "ENTRE "E" AL FINAL D
E LOS DATOS"
1060 INPUT Q$: IF Q$="" THEN GO
TO 1060
1070 IF Q$="E" THEN GO TO 1120
1080 POKE 23692,-1: PRINT Q$
1090 LET TOTAL=TOTAL+VAL Q$
1100 LET CONT=CONT+1
1110 GO TO 1060
1120 PRINT ''"LA MEDIA ARITMETIC
A ES ";TOTAL/CONT
1130 GO TO 9000
1900 REM *************************
2000 REM MEDIA GEOMETRICA
2010 PRINT "MEDIA GEOMETRICA"
```

```
2020 PRINT "PONGA LOS NUMEROS A
LOS QUE SE"
2030 PRINT "DESEA HALLAR SU MEDI
A GEOMETRICA"
2040 PRINT "PONGA "E" AL FINAL D
E LOS DATOS"
2050 LET TOTAL=1
2060 INPUT Q$: IF Q$="" THEN GO
TO 2060
2070 IF Q$="E" THEN GO TO 2120
2080 LET CONT=CONT+1
2090 LET TOTAL=TOTAL+VAL Q$
2100 POKE 23692,-1: PRINT Q$
2110 GO TO 2060
2120 PRINT '"LA MEDIA GEOMETRICA
 ES ";TOTAL↑(1/CONT)
2130 GO TO 9000
2900 REM ************************
3000 REM MEDIA ARMONICA
3010 PRINT "MEDIA ARMONICA"
3020 PRINT "PONGA LOS NUMEROS A
LOS QUE SE"
3030 PRINT "DESEA HALLAR SU MEDI
A ARMONICA."
3040 PRINT "PONGA "E" AL FINAL D
E LOS DATOS"
3050 INPUT Q$: IF Q$="" THEN GO
TO 3050
3060 IF Q$="E" THEN GO TO 3120
3080 POKE 23692,-1: PRINT Q$
3090 LET TOTAL=TOTAL+(1/VAL Q$)
3100 LET CONT=CONT+1
3110 GO TO 3050
3120 PRINT '"LA MEDIA ARMONICA E
S ";1/(TOTAL/CONT)
3130 GO TO 9000
3900 REM ************************
4000 REM FACTORIAL
4010 PRINT "FACTORIAL"
4020 PRINT "INTRODUZCA UN ENTERO
",,"MENOR QUE 34"
4030 INPUT NUM: IF NUM>=34 THEN
GO TO 4030
4040 LET NUM=INT (NUM)
4050 LET A=1
4060 FOR B=1 TO NUM
4070 LET A=A*B
4080 NEXT B
4100 PRINT '"EL FACTORIAL DE ",N
UM;" ES ";A
8900 REM ************************
9000 PRINT '"SELECCIONE EL PROGR
AMA QUE DESEE"
9010 PRINT '"1 - MEDIA ARITMETIC
A"
9020 PRINT "2 - MEDIA GEOMETRICA
"
9030 PRINT "3 - MEDIA ARMONICA"
9040 PRINT "4 - FACTORIAL"
```

```
9050 PRINT "5 - FIN": PRINT
9060 LET A$=INKEY$
9070 IF A$<"1" OR A$>"5" THEN GO
 TO 9060
9080 LET A=VAL A$
9090 IF A=5 THEN STOP
9100 LET TOTAL=0
9110 LET CONT=0
9120 PRINT "**********************
**********"
9130 GO TO A*1000
```

El ordenador puede realizar tareas más complejas. El programa siguiente, por ejemplo, calcula la media, la desviación tipo, el error probable de la media, y la varianza de hasta 20 distribuciones de frecuencias con 240 individuos en cada una. Igualmente puede determinar la diferencia significativa entre dos distribuciones cualesquiera con la ayuda de la prueba T de "Student"

```
  10 REM PRUEBA T
  12 REM DERIVADA DE UN PROGRAMA
         DE ALLAN NORLIN
  13 REM POR ANDERS OLUND
  20 DIM X(240)
  22 DIM N(20)
  24 DIM M(20)
  26 DIM S(20)
  28 DIM E(20)
  30 DIM V(20)
  32 DIM G(20)
 110 PRINT "ESTE PROGRAMA, CALCU
LA LA MEDIA,"
 115 PRINT "LA DESVIACION TIPO,
EL ERROR"
 118 PRINT "PROBABLE Y LA VARIAN
ZA PARA"
 122 PRINT "UN NUMERO DE DISTRIB
UCIONES"
 123 PRINT "DE FRECUENCIAS (MAXI
MO 20"
 124 PRINT "DISTRIBUCIONES), Y L
A PRUEBA"
 130 PRINT "T DE STUDENT."
 135 PRINT "DETERMINA SI HAY UNA
 DIFERENCIA"
 140 PRINT "SIGNIFICATIVA ENTRE
DOS"
 141 PRINT "DISTRIBUCIONES DE FR
ECUENCIAS"
 142 PRINT "CUALESQUIERA."
 144 PRINT
 146 PRINT "DESEA LOS RESULTADOS
 SOLO EN "
 147 PRINT "LA PANTALLA (D) O TA
MBIEN EN"
```

```
 148 PRINT "LA IMPRESORA (P)"
 149 INPUT W$
 150 CLS
 160 LET G=0
 161 LET Q=0
 162 LET S=0
 170 PRINT "CUANTAS DISTRIBUCION
ES?"
 180 INPUT M
 200 LET G=G+1
 205 PRINT
 220 PRINT "DISTRIBUCION DE FREC.
. NUM. ";G
 245 PRINT "CUANTOS INDIVIDUOS?"
 250 INPUT N
 252 PRINT
 255 LET N(G)=N
 260 LET S1=0
 262 LET S2=0                 PULSANDO ENTER
 265 PRINT "ENTRAR LOS DATOS DES
PUES DE       PULSAR NEWLINE"
 268 PAUSE 500
 270 FOR I=1 TO N
 271 CLS
 280 PRINT "NUM. ";I
 290 INPUT X
 295 CLS
 300 LET S1=S1+X
 310 LET S2=S2+X*X
 320 LET X(I)=X
 330 NEXT I
 340 LET M(G)=S1/N
 345 LET V(G)=((S2-S1*S1/N)/(N-1
))
 350 LET S(G)=SQR V(G)
 360 LET E(G)=S(G)/SQR (N)
 380 PRINT "DESEA LA PRESENTACIO
N DE","LOS DATOS?      S/N"
 390 INPUT Q$
 395 CLS
 400 IF Q$="N" THEN GO TO 550
 405 PRINT "DISTRIBUCION DE FREC
. NUM.: ";G
 410 LET Q=Q+1
 415 LET R=1
 417 LET K=0
 420 FOR I=1 TO N
 425 PRINT AT R,K;I;":";X(I)
 430 IF I/20=INT (I/20) THEN LET
 K=K+10
 431 IF K>21 AND W$="P" THEN COP
Y
 432 IF K>21 THEN GO SUB 3000
 435 IF K>21 THEN LET K=K-30
 460 IF I/20=INT (I/20) THEN LET
 R=R-20
 465 LET R=R+1
 480 NEXT I
 490 IF W$="P" THEN COPY
```

```
 500 GO SUB 3000
 550 IF G<M THEN GO TO 200
 740 FOR I=1 TO M
 750 PRINT "DISTRIBUCION DE FREC
. NUM. ";I
 752 PRINT
 755 PRINT "NUMERO DE DATOS =";N
(I)
 758 PRINT
 760 PRINT "MEDIA = ";M(I)
 762 PRINT
 765 PRINT "DESUIACION TIPO = ";
S(I)
 767 PRINT
 770 PRINT "ERROR PROBABLE = ";E
(I)
 772 PRINT
 775 PRINT "VARIANZA = ";E(I)
 776 IF W$="P" THEN COPY
 777 GO SUB 3000
 780 NEXT I
 790 PRINT
 800 PRINT "DESEA LA PRUEBA T?
S/N"
 810 INPUT Q$
 815 CLS
 820 IF Q$="N" THEN GO TO 1200
 830 PRINT "PRUEBA T ENTRE DOS
                DISTRIBUCIONES"
 835 PRINT
 840 PRINT "NUM. ENTRADA DE LA P
RIMERA       DISTRIBUCION"
 845 INPUT S1
 847 PRINT "NUM. ";S1
 849 PRINT
 850 PRINT "NUM. ENTRADA DE LA S
EGUNDA       DISTRIBUCION"
 852 INPUT S2
 854 PRINT "NUM. ";S2
 890 LET A=N(S1)+N(S2)
 900 LET B=N(S1)*N(S2)
 910 LET D=A-2
 920 LET T=ABS (M(S1)-M(S2))/SQR
 (((N(S1)-1)*S(S1)↑2+(N(S2)-1)*S
(S2)↑2)*A/(B*D))
 930 GO SUB 2000
 935 CLS
 940 PRINT "PRUEBA T PARA LAS DI
STRIBUCIONES"
 941 PRINT "NUM. ";S1;" Y NUM. "
;S2
 945 PRINT
 950 PRINT "T=";T
 955 PRINT
 960 PRINT "DF=";D
 965 PRINT
 970 PRINT "P=";P
 975 PRINT
 980 IF P>0.001 THEN GO TO 990
```

```
 982 PRINT "P<0.001    ***"
 984 GO TO 1050
 990 IF P>0.01 THEN GO TO 1000
 992 PRINT "P<0.01    **"
 994 GO TO 1050
1000 IF P>0.05 THEN GO TO 1010
1002 PRINT "P<0.05    *"
1004 GO TO 1050
1010 PRINT "P>0.05  NS."
1050 IF W$="P" THEN COPY
1055 PRINT
1056 PRINT
1057 PRINT
1058 PRINT
1060 PRINT "DESEA PROBAR MAS"
1065 PRINT "DISTRIBUCIONES? S/N"
1070 INPUT Q$
1075 CLS
1080 IF Q$="N" THEN GO TO 1200
1090 GO TO 840
1200 PRINT "DESEA ANALIZAR MAS",
"DISTRIBUCIONES DE FRECUENCIAS?S/N"
1210 INPUT Q$
1215 CLS
1220 IF Q$="S" THEN GO TO 110
1240 PRINT "******  FIN  ******"
1245 GO TO 9999
2000 REM CALCULA VALOR T, VALOR
P (PROBABILIDAD) Y VALOR DF (GRA
DOS DE LIBERTAD)
2020 LET P=1
2025 LET M=1
2030 IF T=0 THEN GO TO 2210
2040 LET U=T*T
2050 IF U<1 THEN GO TO 2100
2060 LET I=M
2070 LET J=D
2080 LET T1=U
2090 GO TO 2130
2100 LET I=D
2110 LET J=M
2120 LET T1=1/U
2130 LET A1=2/9/I
2140 LET B1=2/9/J
2150 LET Z=ABS ((1-B1)*T1↑(1/3)-
1+A1)/SQR (B1*T1↑(2/3)+A1)
2160 IF J>3 THEN GO TO 2180
2170 LET Z=Z*(1+.08*Z↑4/J↑3)
2180 LET P=.5/(1+Z*(.196854+Z*(.
115194+Z*(.000344+Z*.019527))))↑4
2190 IF U>1 THEN GO TO 2210
2200 LET P=1-P
2210 RETURN
3000 PRINT AT 21,0;"PARA CONTINU
AR PULSAR C"
3004 IF INKEY$<>"C" THEN GO TO 3004
3007 CLS
3010 RETURN
9999 STOP
```

La evolución de dos especies

El programa final de esta sección utiliza el ordenador para simular ciclos vitales de dos especies, una de las cuales es depredadora de la otra, y representa gráficamente sus poblaciones relativas. La relación entre las dos especies está determinada por una ecuación diferencial. Se introducen como datos las poblaciones de partida por números comprendidos del uno al cinco. Las fracciones son aceptables y es fascinante entrar en una población muy baja para una de las especies y muy alta para la otra, observando su evolución. Cuando el programa haya analizado un determinado número de generaciones, se entrarán otras poblaciones de partida para las dos especies. El desarrollo de esta relación se representa gráficamente encima del trazado ya existente, por lo que puede obtenerse un número de gráficos mostrando los efectos de las poblaciones de entrada para el depredador y su presa.

```
   5 BORDER 2: PAPER 6: CLS
  10 REM ESPECIES
  20 PRINT "CUANTOS DE LA ESPECI
E UNA?"'"(DE 1 A 5)"
  30 INPUT X: IF X<1 OR X>5 THEN
 GO TO 30
  40 PRINT ''"Y DE LA ESPECIE DO
S?"'"(DE 1 A 5)"
  50 INPUT Y: IF Y<1 OR Y>5 THEN
 GO TO 50
  55 LET X=X+RND: CLS
  60 FOR Z=1 TO 20
  70 FOR T=1 TO 7 STEP .5
  80 PRINT AT 1,1;"ESPECIE UNA:"
; FLASH 1; BRIGHT 1;"  ";INT (X*
10000);"    "
  90 PRINT AT 2,1; INK 1;"ESPECI
E DOS: "; FLASH 1; BRIGHT 1; INK
 2;"  ";INT (Y*10000);"    "
 100 LET X=X+(4*X-2*X*Y)*0.01
 110 LET Y=Y+(X*Y-3*Y)*0.01
 120 PLOT 30*X,30*Y
 130 NEXT T
 135 BEEP .05,Z+T
 140 NEXT Z
 150 LET X=RND*6
 160 LET Y=RND*6
 180 GO TO 60
```

Funciones

El dialecto del BASIC de este ordenador, en común con otras variantes, contiene un número de funciones preprogramadas que pueden utilizarse en un programa o de forma directa. Seguidamente se trata de un programa que utiliza funciones definidas para hacer el dibujo de un murciélago.

Funciones generales:

ABS Esta función, el valor absoluto de una magnitud, proporciona el de X ignorando el signo; de forma que si X era —10,ABS(X) sería 10. Si X era 10,ABS(X) sigue siendo 10.

INT La función INT da el número entero, o la parte entera de una cantidad; su parte mayor no es superior a X. Si es 2.42,INT(X) sería 2.

La función INT redondea las cantidades al número entero más próximo por debajo; ej. INT 2.2 es 2; INT 2.9 es 2; INT 2 es 2, etc. Un requisito frecuente es el redondeo al valor entero más próximo, de forma que 2.6 se haga 3, etc. (algunas instrucciones lo hacen automáticamente; por ejemplo: PRINT AT, POKE). Esto es bastante fácil de hacer. Supongamos que el número que se va a redondear sea A. Si le añadimos primeramente 0.5 y aplicamos la función INT, la respuesta será el número entero más próximo. Si como ejemplo, A fuera 2.6 y deseamos su redondeo a la cantidad entera más próxima: PRINT INT (2.6 + 0.5) nos daría 3, mientras que PRINT INT (2.3 + 0.5) sería 2.PRINT INT (2.5 + 0.5) se hace 3.

Con frecuencia en los cálculos con monedas se necesitan respuestas con dos lugares decimales. Sigue una rutina que hace esto. También incluye un cero delante del punto decimal si el resultado es menor de un dólar ($1.00). Entrese la cantidad como A en la línea 10, en dólares pero sin su signo $, que será añadido por la rutina en la línea 50.

```
 10 INPUT A
 20 LET A$=STR$ (INT (A*100+0.5
)/100)
 25 IF A$(1)="." THEN LET A$="0
"+A$
 30 LET B=LEN A$-LEN STR$ INT V
AL A$
 40 LET A$=A$+(".00" AND B=0)+(
"0" AND B=2)
 50 PRINT "$";A$
 60 GO TO 10
```

```
11.245          $11.24
33              $33.00
33.1            $33.10
333.888         $333.89
0.9999          $1.00
22.009          $22.01
```

RND Función que genera números aleatorios comprendidos entre 0 y 1.

SGN Esta función proporciona el signo de la variable encerrada entre paréntesis, el signo del argumento, que es como se conoce a estas variables. Si X es igual a 20, es decir, X es un número positivo, SGN(X) = 1; SGN(−20) = −1; SGN(0) = 0.

TAB Como se dijo anteriormente en este libro, se trata de una función de tabulación que mueve la posición de la presentación a lo largo de la línea el número de espacios indicados en su argumento. Así, PRINT TAB(6);"$", presentará el signo $ en la séptima posición desde el margen izquierdo: PRINT TAB(13);"$" llevará el signo al décimocuarto lugar. El sentido descendente de la pantalla también puede especificarse añadiendo un segundo argumento detrás de una coma y ambos encerrados entre paréntesis. Así PRINT TAB(4,9);$ hará la presentación de un signo del dólar cinco espacios abajo y diez en sentido transversal. TAB reduce en 32 un módulo numérico, significando que el argumento del número que sigue a TAB puede ser mayor que 31; y en este caso lo reducirá hasta dejarlo comprendido entre 0 y 31, moviéndose el lugar de la presentación en la misma línea a menos que ello implique retro-

ceso, en cuyo caso se mueve a la línea siguiente. Esta cuestión del módulo significa que el argumento de TAB se divide por 32 (el número de columnas por líneas de la pantalla) y se toma el resto. Puede aprovecharse esta circunstancia cuando el espacio PRINT sea determinado por un cálculo ya que no es preciso asegurar que la cantidad determinada quede en el margen entre 0 y 31.

EXP Esta función proporciona el valor de e elevado a la potencia del argumento; así si se escribe: PRINT EXP 5 nos dará 148.41316.

LN LN X nos da el logaritmo neperiano de X; PRINT LN 5 = = 1.6094379.

SQR Es la raíz cuadrada de un número. Si X = 5; PRINT SQR X = = 2.236068.

Funciones trigonométricas

SIN Es el SENO de un ángulo en radianes. SIN 5 = 0,95892428.

COS Proporciona el COSENO de un ángulo en radianes. PRINT COS X, si X = 5; nos dará 0.28366219.

TAN Produce la tangente del ángulo X en radianes.

(El ordenador mide los ángulos en radianes. PI radianes equivalen a 180 grados.)

La función RANDOMIZE actúa como sigue:

El número que se coloca detrás de la palabra RANDOMIZE se conserva en las variables del sistema después de haber sido redondeado al entero más próximo. Si solamente se aplica la función sin argumento o con argumento 0 proporciona el valor del contador de cuadros para la formación de imagen en pantalla. Este valor no es afectado por los enunciados CLEAR o RUN pero se hace 0 con el NEW. Cambia cada vez que se utiliza RND.

Conversión de otros dialectos del BASIC

Existen en muchos libros y revistas especializadas una gran riqueza de programas redactados en BASIC pero, dado que todas las versiones de este lenguaje difieren en algo, no es probable que un programa escrito para otro ordenador pueda ejecutarse en el propio sin algunos cambios. La extensión y naturaleza de estos cambios dependen en gran manera de la estructura del programa en particular y de cómo maneja los datos pero es posible dar algunas normas generales sobre qué aspectos hay que considerar cuando uno se enfrente con la tarea de convertir un programa "ajeno" en otro asimilable por nuestro ordenador.

La aritmética de números enteros

Añádase siempre, en general, la función INT delante de una división en los programas diseñados para ordenadores con aritmética de números enteros. Puede ser preciso el uso de paréntesis en la división para que la citada función INT sólo trabaje sobre su resultado.

La sentencia DIM

Algunos dialectos del BASIC permiten escribir varias sentencias DIM en una misma línea, como en DIM A$(9), B$(8), C$(7). Habrá que substituir esto por sentencias DIM individualizadas en líneas separadas del programa. Si éste trabaja con conjuntos cuyos nombres

tienen más de una letra, tienen que ser sustituidos por otros que tengan solamente una, como A$ o B. Si no se dispone de letras suficientes habrá que declarar dimensiones adicionales a las existentes para un cierto conjunto y utilizar la dimensión incorporada para substituir uno de tales conjuntos. Téngase cuidado con los subíndices 0.

Funciones GET Y GET$

Leen caracteres o valores de las teclas pulsadas en el teclado. Toman varias formas en diversos ordenadores pero, en general, esperan hasta que se pulsa la tecla antes de actuar, asignando el carácter correspondiente a la pulsación o el código de dicho carácter a una variable. Por ejemplo, GET A$ o LET A$ = GET$. Puede hacer lo siguiente en su ordenador:

```
1000 LET A$=INKEY$
1010 IF A$="" THEN GO TO 1000
```

Esto regresaría el carácter correspondiente a la tecla pulsada. Si la función fuese recoger el código (CODE) (probablemente escrito como ASC) del carácter, utilícese entonces esta rutina:

```
1000 LET A$=INKEY$
1010 IF A$="" THEN GO TO 1000
1020 LET A=CODE A$
1030 PRINT A
```

Ligeramente diferente es la versión que recoge un valor numérico en lugar del código del carácter. Es necesario asegurarse que el carácter leído del teclado se encuentra en la escala comprendida entre el "0" y el "9", de forma que pueda aplicarse la función VAL para convertir el carácter en un número. He aquí una forma de hacerlo:

```
1000 LET A$=INKEY$
1010 IF A$<"0" OR A$>"9" THEN GO
 TO 1000
1020 LET A=VAL A$
1030 PRINT A
```

Es posible encontrar también una versión de la función INKEY$ que permite un límite de tiempo a especificar para las respuestas del usuario; ej.:

100 LET A$ = INKEY$(X)

donde X especifica el límite de tiempo. Esto puede convertirse de dos formas. Una:

```
 10 LET X=50
100 PAUSE X
110 LET A$=INKEY$
```

o como sigue, demostrado con un simple juego:

```
   5 REM   PONGA PRIMERO LA FIJA
CION DE MAYUSCULAS
  10 LET B$=CHR$ (INT (RND*26) +C
ODE "A")
  20 PRINT AT 10,0;"RAPIDO, PULS
A ";B$
 100 FOR A=0 TO 100
 110 LET A$=INKEY$
 120 IF A$<>"" THEN GO TO 140
 125 NEXT A
 130 PRINT "TIEMPO CONSUMIDO": S
TOP
 140 IF A$=B$ THEN PRINT  "HAS A
CERTADO": STOP
 150 PRINT "TE HAS EQUIVOCADO"
```

La función VAL

Si el argumento de VAL no forma uno numérico válido se produce un mensaje de error. En otros dialectos del BASIC retorna a 0.

Las funciones SET,RESET

Se utilizan para producir un punto blanco o negro en la pantalla. Sustitúyanse por PLOT/OVER/PRINT AT.

La función ELSE

Es una extensión de la función IF...THEN, sentencia condicional, y permite más de una alternativa, dependiendo de que la instrucción condicional sea verdadera o falsa. Puede ser sustituida por dos expresiones condicionales en su ordenador. Por ejemplo:

```
 20 IF X=1 THEN LET Y=7 ELSE GO
TO 80
```

puede ser substituida por:

```
20 IF X=1 THEN LET Y=7
21 IF X<>1 THEN GO TO 80
```

Si la acción de ELSE es asignar uno de los varios valores alternativos a una variable, puede sustituirse con una línea. Ej.:

```
50 IF X=1 THEN LET Y=7 ELSE LET
Y=8
```

se sustituye por:

```
 50 LET Y=(7 AND X=1)+(8 AND X<
>1)
```

Ciertas expresiones como las tratadas anteriormente pueden ser sustituidas por formas más cortas tales como:

```
50 LET Y=7+(1 AND X<>1)
```

No puede darse ninguna norma general ya que el método utilizado varía según el caso de que se trate; los ejemplos anteriores dan una idea de cómo proceder.

Podemos encontrarnos con una instrucción en la que la acción realizada por la función ELSE sea condicional en sí misma:

```
10 IF X=1 THEN LET Y=1 ELSE IF X
=5 THEN GO TO 100
```

Esto puede resolverse así:

```
 10 IF X=1 THEN LET Y=1
 11 IF X<>1 THEN IF X=5 THEN GO
TO 100
```

o bien:

```
 10 IF X=1 THEN LET Y=1
 11 IF X<>1 AND X=5 THEN GO TO
100
```

Es posible encontrar toda clase de instrucciones condicionales ELSE y las versiones del Spectrum dependerán de las variaciones encontradas.

REPEAT... UNTIL

Este es un bucle que realiza una operación continuamente, terminando sólo cuando se cumple una determinada condición. Su utilización es tan extendida que es difícil especificar un método universal de conversión al BASIC del Spectrum; siendo el mejor probablemente la instrucción condicional IF...THEN GO TO. He aquí un ejemplo:

```
10 PRINT "PONER SI O NO"
20 REPEAT
30 INPUT A$
40 UNTIL A$="SI" OR A$="NO"
```

puede sustituirse por:

```
 10 INPUT "DECIR SI O NO ";A$
 20 IF A$<>"SI" AND A$<>"NO" TH
EN GO TO 10
```

Las estructuras de REPEAT... UNTIL son generalmente mucho más complejas que en este ejemplo y puede ser preciso encontrar medios de conversión distintos al IF...THEN GO TO. Por ejemplo, cuando el valor de una variable es el factor determinante, puede utilizarse a veces un bucle FOR/NEXT. Sin embargo, la posibilidad de utilizar una proposición condicional IF...THEN GO TO debe considerarse siempre y es, en ocasiones, el único método aceptable de conversión.

Variables no definidas

Si se intenta utilizar una variable antes de haber sido definida o asignada en un programa, algunos ordenadores pondrán su valor en 0. En el Spectrum se produce un mensaje de error en este caso. En consecuencia, todas las variables utilizadas en su ordenador han de estar asignadas.

Matrices

Algunos dialectos del BASIC tienen funciones matriciales que realizan operaciones sobre conjuntos. Su ordenador no tiene estas funciones por lo que habrá que realizar individualmente las operaciones con los elementos del conjunto, posiblemente por medio de un bucle.

```
10 DIM X(Y)
20 DIM P(Y)
30 MAT X=P
```

Este ejemplo particular puede ser sustituido por:

```
10 LET N=0
20 DIM X(Y)
30 DIM P(Y)
40 LET N=N+1
50 IF N<Y THEN GO TO 40
```

PROC, ENDPROC

Este es un método de utilizar las subrutinas para emplear ciertos procedimientos de tal forma que, entre otras cosas, hace los programas y los listados más fáciles de comprender y de leer (algunos llaman a estas técnicas *programación estructurada*). Permiten que las subrutinas se utilicen específicamente para hacer ciertas cosas aunque con la importante excepción de que son requeridas por su nombre en lugar de por los números de línea. Tómese este ejemplo que presenta una puntuación en la pantalla:

```
100 PROCpuntos

........
1000 DEF PROCpuntos
1010 PRINT "PUNTOS=";S
1020 ENDPROC
```

La función ENDPROC es, de alguna manera, similar a RETURN en que una vez el procedimiento se ha completado, el programa continúa desde la línea inmediata a la que la ha requerido, en este caso la de después de la 100. El nombre de referencia que incorpora PROC no existe en la versión análoga que lo sustituye en el Spectrum pero puede adoptarse una solución como se ve en el segundo ejemplo. El método más sencillo de conversión al BASIC del ZX es por la línea 100 GOSUB 1000, con una instrucción REM en cualquier parte para identificar la subrutina y el final con la instrucción RETURN.

```
100 GO SUB 1000

......
1000 REM SUBRUTINA PUNTUACION
1010 PRINT "PUNTUACION=";S
1020 RETURN
```

Si se desea retener la denominación del procedimiento o subrutina puede utilizarse una variable con el mismo nombre que el asignado al enunciado PROC durante el curso del programa antes de requerirse la subrutina y utilizar esta variable como destino de la instrucción

GOSUB (ir a subrutina). Puede incluirse una REM para identificar la subrutina y relacionarla con el nombre de la variable utilizada. Es útil usar caracteres inversos en estas REM para que puedan destacar del resto del texto listado. Así se puede hacer que los programas parezcan bastante estructurados.

```
50 LET PUNTUACION=1000
......
100 GO SUB PUNTUACION
......
1000 REM SUBRUTINA PUNTUACION
1010 PRINT "PUNTUACION=";S
1020 RETURN
```

Aunque el enunciado PROC puede ser complejo, una subrutina es el mejor sistema para la conversión al BASIC del Spectrum (las instrucciones GOSUB/RETURN).

La función INSTR(A$, B$)

Esta es una función que investiga si hay una copia de B$ en A$ y, si es así, dice dónde empieza dicha copia. Por ejemplo, si B$ fuera "PUT" y A$ "COMPUTER", el valor de INSTR(A$,B$) sería 4, debido a que la parte de A$ que contiene las letras "PUT" empieza en el cuarto elemento de A$. Si la función no encuentra la copia buscada, INSTR$(A$,B$) daría 0. Hay que preparar una rutina especial para conseguir esta función en el ZX Spectrum.

He aquí un método de conversión adecuado:

```
HAPPY P
INSTR(A$,B$)=3
HAPPY H
INSTR(A$,B$)=1
HAPPY SAD
INSTR(A$,B$)=0
SINCLAIR BASIC
INSTR(A$,B$)=0
SINCLAIR SIN
INSTR(A$,B$)=1
```

```
  10 REM -- LET Y=INSTR(A$,B$)--
  20 INPUT A$
  30 INPUT B$
  40 PRINT A$;"  ";B$
  50 GO SUB 1000
  60 PRINT "INSTR(A$,B$)=";Y
  70 GO TO 20
1000 REM SUBRUTINA PARA 'INSTR'
1010 LET Y=0
1020 IF LEN A$=0 OR LEN B$=0 OR
LEN B$>LEN A$ THEN RETURN
1030 FOR Y=1 TO LEN A$-LEN B$+1
1040 IF A$(Y TO Y+LEN B$-1)=B$ T
HEN RETURN
1050 NEXT Y
1060 LET Y=0
1070 RETURN
```

Obsérvese que si se desea detectar palabras enteras en lugar de sólo cadenas alfanuméricas, habrá que examinar A$ en busca de espacios o signos de puntuación que indiquen el principio o final de las palabras. La rutina anterior solamente busca cadenas adaptadas, de forma que si se desea encontrar la palabra PAN en una frase que contiene una tal como EXPANSION, se detectarían las letras de aquélla incluidas en ésta. Sin embargo, también los usuarios de la variación INSTR tienen este problema, por lo que de cualquier manera el programa tendrá que ofrecer la solución.

DIV

DIV da la parte entera del resultado de una división; por ejemplo, 17 DIV 5 da 3. Como adaptación podemos utilizar la instrucción INT al resultado de la división en el Spectrum. Así A DIV B sería INT (A/B).

MOD

MOD nos da el *resto* de una división. Por ejemplo: 17 MOD 5 es 2. La adaptación al Spectrum de A MOD B sería A — (INT[A/B*B]). Obsérvese que la función TAB lleva su propia acción MOD (módulo 32) en nuestro ordenador.

La función TAB

Algunos ordenadores pueden tener dos argumentos para la función TAB que sirven para el espaciado en la pantalla. Esta utilización del TAB es equivalente al AT del Spectrum. Por ejemplo, TAB (X,Y) en algunos ordenadores se corresponde con el AT Y,X;del Spectrum. Las coordenadas X e Y pueden tener el orden invertido en algunos ordenadores.

Grados y radianes

Su Spectrum efectúa las funciones trigonométricas en radianes a los que se pasa a partir de los grados sexagesimales por medio de la siguiente expresión:

LET RADIANES = (PI*GRADOS)/180

y los radianes se conviertan en grados por:

LET GRADOS = (180*RADIANES)/PI

Logaritmos de base 10

Su ordenador trabaja con logaritmos naturales, de base **e**, y si se precisan por cualquier razón los de base 10 han de calcularse mediante la expresión:

LET LOGBASE10 (X) = (LN[X])/(LN[10])

Puede utilizarse igualmente para logaritmos de cualquier base. Supongamos que se necesita el $\log_B X$:

LET LOGBASEB (X) = (LN[X])/(LN[B])

El signo de porcentaje (%)

Se utiliza este símbolo para expresar una variable entera. Por ejemplo: A%. Se utilizan principalmente para ahorrar memoria o debido a que pueden ser procesadas más de prisa que las variables convencionales. En general, no hay ningún inconveniente en utilizar una variable ordinaria aunque se debe ser cauto con estas variables enteras al ser asignadas si proceden del resultado de una división ya que automáticamente redondean el cociente a su valor entero. En este caso debe utilizarse LET A = INT (A/2), por ejemplo, para hacer entero el resultado de la división.

Signo de interrogación (?)

En la mayoría de los ordenadores el símbolo ? se utiliza como abreviatura del enunciado PRINT.

PEEK y POKE

Estas son dos instrucciones muy poderosas que permiten hacer cosas con el ordenador que no serían posibles de otra manera. Empecemos por definir los dos términos PEEK y POKE.

(1) PEEK m nos da el número recogido en la dirección m de la memoria.
(2) POKE m,n pone el número n en la dirección m de la memoria. Cuando es aceptado se borra el que allí había.

El término dirección ("address") necesita explicación. Un ordenador como el Spectrum piensa y recuerda por medio de números, no

con palabras como hace la gente. Ciertos conjuntos de números hacen que alguna parte del ordenador realice cosas específicas. Esto se denomina programa. Ahora bien, se necesita un procedimiento para conservar todos estos números hasta que se les necesite y una vez se conocen sus valores y su distribución pueda el ordenador decidir lo que va a hacer.

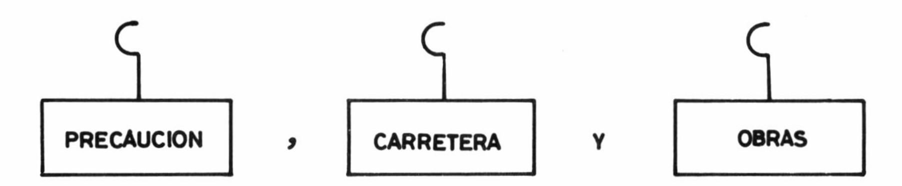
Cierta disposición de los números puede hacer que el ordenador presente algo en la pantalla, sume dos cantidades o quizá que interrumpa el programa si aquella disposición de los números pretende hacer algo que no se puede o no se debe.

El ordenador no puede poner simplemente los números en cualquier parte. Sería un caos si no supiera dónde ha de mirar después de cada paso. En consecuencia, existe un método por el que todo queda perfectamente organizado.

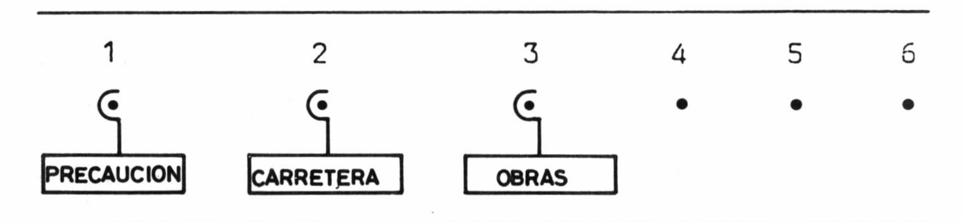
Imagínese que se desea presentar un mensaje al público y se dispone de las palabras escritas en pequeños carteles con ganchos preparados para cualquier caso, de forma que es posible poner de manifiesto el mensaje colgando el adecuado juego de los mismos. Por ejemplo, si deseamos presentar el siguiente mensaje:

"PRECAUCION CARRETERA OBRAS",

necesitamos los carteles que siguen:



Se precisará igualmente un tablero donde colgar los carteles con estas palabras. Si empezamos con el primer clavo colgando en él la primera palabra y seguimos por orden acabaremos con un mensaje claro:



Los clavos del tablero nos dicen dónde se cuelga cada palabra. Esto es una buena comparación con la memoria de su ordenador. Existen 65536 lugares donde se pueden colgar los números del Spectrum; tales lugares están divididos para varios usos y tanto usted como el ordenador pueden hacer varias cosas con ellos. Estos ''clavos'' o localizaciones, o como los quiera llamar, se denominan realmente direcciones (''addresses'') (el hogar de cada número, si se desea). Sin embargo, si el Spectrum tiene un número que desea guardar, no puede simplemente meterlo en cualquier parte porque podría trastornar lo que hay allí.

Una de las formas en las que ya usted ha utilizado el POKE es para crear gráficos definidos por el usuario. Ellos han sido introducidos (POKEd) en las direcciones que empezaban en la 32600 en un Spectrum de 16K.

Si miramos en el primer ''clavo'' (PEEK 1) encontramos la palabra PRECAUCION allí. Si lo hacemos en el segundo (PEEK 2) encontraremos la palabra CARRETERA, y así sucesivamente. ¿Puede verse la analogía? Recuérdese que el ordenador utiliza números en lugar de palabras pero la idea sigue siendo la misma. De forma similar podemos cambiar las palabras de los ''clavos'' muy fácilmente utilizando el enunciado POKE para poner un nuevo número donde solía estar otro. Podríamos hacer algo así como introducir (POKE) en el ''clavo'' 2 la palabra EDIFICIO que pondría este vocablo en el segundo lugar del letrero, cambiando enteramente su significado. El gran secreto sobre PEEK y POKE no está en lo que hacen sino en cómo utilizarlos. Está muy bien encontrar qué número se encuentra en una determinada dirección pero ¿cómo puede hacerse uso de esto en un programa y cómo saber dónde mirar (PEEK) y meter (POKE)? La respuesta es: esencialmente con la experiencia y leyendo los programas de otros, aunque se encontrará que, a medida que crece el conocimiento sobre los ordenadores, se apreciará cómo se descubren nuevas formas de utilizar PEEK y POKE. Antes de pasar a los ejemplos haremos un breve recordatorio de cómo escribir las instrucciones PEEK y POKE.

PEEK m. La dirección en la que estamos mirando es m, y además es un número comprendido entre 0 y 65560; m también puede ser el resultado de un determinado cálculo. POKE m,n. Aquí m es una dirección donde se va a poner el nuevo número, como en PEEK. Se escribe entre la palabra POKE y la coma. El número tras la coma, n, es el que se va a colocar en la dirección m y puede ser cualquiera comprendido entre 0 y 255, o el resultado de un cálculo, siempre que esté

comprendido en el citado intervalo. Se puede hacer que n sea negativo, igualmente entre 0 y −255, aunque esto se hace raramente y no es muy útil.

Veamos ahora algunos ejemplos de la utilización de PEEK y POKE.

(1) Instrucciones REM

Muchos programas se apoyan en la información incluida en las instrucciones REM en la primera línea de un programa en BASIC. Esto es así porque es fácil de introducirse y muy económico de memoria. El punto importante es que la dirección del primer carácter después de la palabra REM en la primera línea de un programa es 23760. En consecuencia, si se tiene el programa:

```
1 REM ABCDEF
2 PRINT PEEK 23760
```

se presentaría el número 65 en la pantalla. Este es el código (CODE) del carácter A, así que la dirección 23760 tiene el valor de 65.

Puede fácilmente cambiarse este valor situando otro nuevo en 23760. Por ejemplo, para cambiar A por Z, búsquese el código de Z, que es 90, y colóquese en la dirección 23760:

```
POKE 23760,90
```

O podría haberse escrito POKE 23760,CODE "Z" que hace exactamente lo mismo. La siguiente dirección, 23761 acoge la B;23762, la C, y así sucesivamente.

La técnica de "mirar" y "meter" (PEEKING/POKING) en las instrucciones REM fue de gran importancia para el almacenaje de los programas en código máquina del ordenador ZX81.

Aunque todavía puede utilizarse, el manual (en la sección denominada "Usando el Código Máquina" explica una forma de reservar memoria e introducir (POKE) el código máquina en el área reservada.

(2) La utilización del reloj

El reloj está contenido en las direcciones 23672, 23673 y 23674. Simplemente cuenta el número de cuadros enviados al televisor. Puede usted "mirar" (PEEK) y "meter" (POKE) en estas direcciones.

Sitúese de nuevo el contador a cero con estas instrucciones:

POKE 23674,255
POKE 23673,255
POKE 23672,255

Y para leer su valor utilícese esta expresión:

```
65536* PEEK 23674 + PEEK 23672 +
 256* PEEK 23673
```

Esto nos da una respuesta en cuadros y puesto que 50 cuadros se envían cada segundo al televisor (60 en Estados Unidos), se precisa dividir por 50 (ó 60) para que nos venga en segundos, como lo que sigue:

```
LET TIME=(65536*PEEK 23674+PEEK
23672+256*PEEK 23673)/50
```

He aquí un programa para proporcionar un cronómetro:

```
  5 POKE 23674,255
 10 POKE 23673,255
 20 POKE 23672,255
 30 LET TIEMPO=(65536*PEEK 2367
4+PEEK 23672+256*PEEK 23673)/50
 40 PRINT AT 11,14;INT (TIEMPO*
10)/10;"     "
 50 GO TO 30
```

La instrucción INT en la línea 40 se añade para impedir la presentación de fracciones de segundo inferiores a la décima. Este cronómetro mide el tiempo con bastante precisión ya que el contador de cuadro está controlado con equipo especial, de forma que, a menos que se fuerce deliberadamente el programa a actuar de otra manera, su funcionamiento es independiente de la velocidad de ejecución y cuenta el tiempo bastante bien. Las posibilidades del contador de cuadro permite la duración del medidor durante casi cuatro días. Si se desea la presentación de minutos y segundos utilícese esta rutina:

```
  5 POKE 23674,255
 10 POKE 23673,255
 20 POKE 23672,255
 30 LET TIEMPO=(65536*PEEK 2367
4+PEEK 23672+256*PEEK 23673)/50
 40 PRINT AT 11,12;INT (TIEMPO/
60);":";INT (TIEMPO-INT (TIEMPO/
60)*60);"     "
 50 GO TO 30
```

Aplicaciones mercantiles

Este ordenador puede utilizarse en ciertas aplicaciones para pequeños negocios. Existe una amplia variedad de programas para explotar su gran potencial. En esta sección analizaremos algunos programas de sencilla aplicación práctica del Spectrum de Sinclair.

El primero es una manipulación de dinero. James Walsh ha escrito un programa para el cálculo del interés compuesto. Las notas para el usuario son claras y fáciles de seguir.

```
   90 LET A$="ANUALIDAD   INTERESE
S     TOTAL"
  100 INPUT "NUMERO DE ANUALIDADE
S? ";A
  110 PRINT "INTERES COMPUESTO"
  120 PRINT ''"SOBRE ";A;" ANUALI
DADES"
  130 INPUT "CANTIDAD? ";C: LET T
=C
  140 PRINT ''"EL CAPITAL ES ";C;
" PTS."
  150 INPUT "INTERES POR ANUALIDA
D? ";IN
  160 CLS : PRINT AT 1,0;
  170 FOR N=1 TO A
  180 POKE 23692,-1
  200 GO SUB 340
  210 PRINT N;TAB 10;"PTS.";INT (
V+.5);TAB 21;"PTS.";INT (T+.5)
  260 NEXT N
  270 PRINT '"TOTAL= ";INT (T+.5)
;" PTS."
  290 PRINT '"INTERESES= ";IN;"%"
  310 PRINT '"CANTIDAD ORIGINAL=
";C;" PTS."
  320 PRINT AT 0,0;A$
  330 STOP
  340 LET V=1+(IN/100)*T
  350 LET T=(IN/100)*T+T
  360 RETURN
```

Tratamiento de textos

Este programa dejará el texto limpio y correcto antes de imprimirlo y proporciona la oportunidad de corregir los errores utilizando un cursor de movimiento libre. Se escribe el texto hasta 17 líneas como una sola cadena, X$. Cuando se tiene el texto introducido, se pulsa ENTER y el ordenador dispondrá las palabras para asegurar que ninguna de ellas quede dividida al final de una línea.

Aparece un repertorio (menú) con tres opciones: 1, corrección del texto; 2, impresión (LPRINT); y 3, empezar nuevamente. Si se decide corregir el texto, reaparecerá en la pantalla con el mensaje: "PONGA 1 PARA REGRESAR AL MENU" escrito en la parte superior de dicho texto. Se utilizan las teclas 5, 6, 7 y 8 para mover el cursor en la dirección indicada por las flechas de esas teclas haciendo que se desplace a lo largo de las líneas del texto y poniendo un cuadradito negro sobre la letra por la que se está pasando. Una vez que se encuentra una equivocada se pulsa la "A" y aparece el mensaje "ponga letra" (entre letter) en el fondo de la pantalla. Se pone tal letra y se acciona ENTER, con lo que se cambia a la seleccionada. Pulsando "1" en cualquier momento se pasa de la fase de corrección al repertorio (menú) original y de ahí se puede escoger el "2" para poner el texto en la impresora.

Después de la impresión se muestra otro repertorio (menú) que permite pasar nuevamente todo el programa con el borrador establecido o terminar su ejecución.

```
  10 REM TRATAMIENTO DE TEXTO
  20 PRINT "PONGA EL TEXTO"
  30 INPUT X$
  32 LET X$=X$+"
 "
  35 CLS
  40 GO SUB 1000
  45 GO SUB 1000
  50 PRINT X$
  70 PRINT '" PONGA 1 PARA CORRE
GIR,             2 PARA IMPRI
MIR,             3 PARA EMPEZ
AR OTRA VEZ"
  80 INPUT Q
 100 IF Q=3 THEN RUN
 110 IF Q=2 THEN GO TO 4000
 120 IF Q=1 THEN GO TO 2000
 130 GO TO 80
1000 REM SE DETIENE EN PALABRAS
DIVIDIDAS
```

```
1010 LET N=1
1020 GO SUB 1180
1030 LET N=N+33
1040 IF N>=LEN X$ THEN RETURN
1045 REM UN ESPACIO EN LA LINEA
SIGUIENTE
1050 IF X$(N)=" " THEN GO TO 116
0
1065 REM UN ESPACIO EN LA LINEA
SIGUIENTE
1070 IF X$(N)=" " THEN GO TO 103
0
1080 LET J=0
1090 GO SUB 1180
1100 LET J=J+1
1105 REM UN ESPACIO EN LA LINEA
SIGUIENTE
1110 IF X$(N)>" " THEN GO TO 109
0
1120 FOR N=N TO N+J-1
1125 REM UN ESPACIO EN LA LINEA
SIGUIENTE
1130 LET X$=X$(1 TO N)+" "+X$(N+
1 TO )
1140 NEXT N
1150 GO TO 1030
1160 LET X$=X$(1 TO N-1)+X$(N+1
TO )
1170 GO TO 1020
1180 LET N=N-1
1190 RETURN
2000 REM * CORRECCIONES *
2005 CLS
2010 CLS
2020 PRINT "PONER 1 PARA REGRESA
R AL MENU"
2030 LET A=1: LET L=LEN X$
2032 LET Z$=X$(1)
2035 PRINT AT 2,0;X$
2037 LET X$(A)=Z$
2050 IF INKEY$="8" AND A<L THEN
LET A=A+1
2055 IF INKEY$="6" AND A<L+32 TH
EN LET A=A+32
2060 IF INKEY$="5" AND A>1 THEN
LET A=A-1
2065 IF INKEY$="7" AND A>32 THEN
 LET A=A-32
2070 IF INKEY$="1" THEN GO TO 70
2080 IF INKEY$="A" THEN GO SUB 3
000
2090 PRINT AT 1,0;A;" ";X$(A);"
  "
2100 LET Z$=X$(A)
2110 LET X$(A)="█"
2120 GO TO 2035
3000 INPUT INK 2; FLASH 1; BRIGH
T 1;"    PONER LETRA    ";H$
3010 LET X$(A)=H$
```

```
3030 RETURN
4000 LPRINT X$
4005 CLS
4010 PRINT "PONER 1 PARA NUEVA I
MPRESION"
4020 PRINT TAB 6;"2 PARA NUEVO P
ASE"
4030 PRINT TAB 6;"3 PARA DETENER
 EL PROGRAMA"
4040 INPUT U
4050 IF U=1 THEN GO TO 4000
4060 IF U=2 THEN RUN
4070 IF U=3 THEN STOP
4080 GO TO 4040
```

El programa final de esta sección tiene por objeto poner entradas y referencias de página en orden alfabético y permite la construcción de índices de libros y de artículos de revistas, pudiendo adaptarse fácilmente para la acomodación de listas o niveles de almacén.

El programa tiene tres partes. La primera (hasta la línea 100) acepta las entradas; la segunda (líneas 200 a la 300) las clasifica, y la tercera (de la 310 a la 530) presenta los datos.

El programa pide que se ponga un título (T$) y su autor (N$) y a continuación introduce los temas y las páginas, una por una, poniéndose una "E" para terminar el proceso de entradas. El programa aceptará hasta 400 entradas (línea 20) en un Spectrum de 16 K; y 2000 o más en uno de 48K. Al final, se puede escoger entre la presentación en pantalla o en la impresora.

He aquí un ejemplo:

```
JUEGOS DE INTELIGENCIA

FISHER R B

APRENDIZAJE - 114
CONOCIMIENTO - 189
INTELIGENCIA - 7
MAQUINARIA - 32
MEMORIA - 92
PERCEPCION - 187
PERSONALIDAD - 9
```

```
  10 REM INDICE DE LIBRO
  20 DIM A$(400,20)
  30 INPUT "PONER EL TITULO ",T$
  40 INPUT "PONER EL NOMBRE DEL
AUTOR ";N$
  50 FOR G=1 TO 500
```

```
  60 INPUT "PONER PALABRA Y NUME
RO DE PAGINA PONER "E" AL FINAL
DE LAS           ENTRADAS ";A$(G)
  70 REM 19 ESPACIOS EN LA LINEA
 SIGUIENTE
  80 IF A$(G)="E
   " THEN GO TO 200
  90 PRINT A$(G)
 100 NEXT G
 200 CLS
 210 PRINT "PREPARADO, CLASIFICA
CION"
 220 FOR B=1 TO G-1
 230 FOR C=B+1 TO G-1
 240 IF A$(B)<=A$(C) THEN GO TO
280
 250 LET D$=A$(B)
 260 LET A$(B)=A$(C)
 270 LET A$(C)=D$
 280 NEXT C
 290 NEXT B
 300 PRINT ''"PREPARADO"
 310 PRINT "PONER 1 PARA IMPRIMI
R LA LISTA"
 320 PRINT '"PONER 2 PARA PRESEN
TACION EN      PANTALLA"
 330 IF INKEY$="2" THEN CLS : GO
 TO 440
 340 IF INKEY$="1" THEN GO TO 36
0
 350 GO TO 330
 360 LPRINT T$
 370 LPRINT
 380 LPRINT N$
 390 LPRINT
 400 FOR A=1 TO G-1
 410 LPRINT A$(A)
 420 NEXT A
 430 STOP
 440 PRINT 'T$
 450 PRINT 'N$
 460 PRINT
 500 FOR A=1 TO G-1
 520 PRINT A$(A)
 530 NEXT A
```

Cómo mejorar los programas

Usted habrá pasado probablemente por varias fases mientras se desarrollaba su habilidad para la programación. Al principio, una breve lucha con el BASIC y súbitamente descubrió que podía, de

alguna manera, escribir programas que funcionaban. Sus listados seguramente presentaban un aspecto bastante enrevesado y sus amigos probablemente necesitaban que usted les diera una explicación detallada antes de que supieran cómo ejecutar sus programas, aunque al final acababan funcionando.

Después llegó la fase en la que se tomó la decisión de que era posible hacer algo más. Pero si bien pudiera sentirse vagamente insatisfecho con sus programas es probable también que no tuviera muy claro cómo mejorar su calidad. He aquí unas cuantas normas que pueden ayudarle.

Primero, eche una ojeada a la impresión de su listado. Los programas enlazados con instrucciones REM tienen mejor aspecto y son más fáciles de entender cuando se vuelve a ellos después de una interrupción. Por supuesto que la escasez de memoria puede impedir el lujo de utilizar REMs pero, si dispone de ella, deben incluirse. Un REM con sólo una línea de asteriscos es muy útil para separar las secciones principales del programa. Examínese críticamente cualquier instrucción GO TO. Demasiados GO TOs saltando como ranas entre distintas partes del programa muestran falta de pensamiento directo, hacen que se ejecuten con más lentitud y son muy difíciles de descifrar.

Es una buena práctica de programación poner cada una de las secciones principales del programa en subrutinas separadas (una para asignar las variables al principio, otra que prepara la presentación, la que determina quién ha ganado y así sucesivamente). El principio pudiera muy bien ser así:

```
   10 REM * NOMBRE DEL PROGRAMA *
   20 REM ASIGNACION DE VARIABLES
   30 GO SUB 9000
   40 REM PREPARACION DE LA PRE
SENTACION
   50 GO SUB 8000
   60 REM MOVIMIENTOS PERSONALES
   70 GO SUB 7000
   80 REM MOVIMIENTOS DEL ORDENA
DOR
   90 GO SUB 6000
  100 REM COMPROBAR SI EL JUEGO
TERMINA
  110 GO SUB 5000
  120 GO TO 50
```

Como puede verse esto asegura que el programa se mueva realmente en un ciclo continuo una y otra vez hasta que termine con la subrutina "COMPROBAR SI EL JUEGO HA TERMINADO". Es posi-

ble escribir una serie de líneas como estas antes de empezar a hacer otra cosa, incluso antes de que sepa cómo va a realizar realmente las tareas de la subrutina.

A continuación puede construirse el programa módulo a módulo, asegurándose de que cada uno funciona antes de pasar al siguiente. Es bastante fácil corregir un programa preparado así y es mucho más sencillo conservar una idea de dónde se encuentra cada cosa si se sigue esta pauta que si se deja que el programa se vaya haciendo, más o menos, a sí mismo.

El listado ha de ser tan transparente como sea posible, tanto para el proceso actual de corrección como para saber en el futuro qué es lo que hace cada cosa. La presentación del programa tiene que ser también buena. Si no hay problema de memoria, asegúrese de que la presentación en pantalla es clara y definida. Usense líneas PRINT en blanco para el debido espaciado, juéguese con las reglas de los símbolos gráficos para dividir la pantalla en secciones lógicas, etc. Una vez que el programa funciona satisfactoriamente, vale la pena emplear algún tiempo en la subrutina que controla la presentación. Aquí nuevamente se apreciará la ventaja de tener toda ella manejada por una subrutina, ya que se sabrá fácilmente dónde acudir para mejorarla.

Dado que no vivimos en un mundo ideal, es improbable que cada enunciado de la presentación pueda estar contenido en una sola subrutina pero si se hace todo lo posible para conseguirlo será fácil trabajar con el programa.

El enfoque "estructurado" que se ha descrito también ayuda a que se comprenda otro objetivo de un buen programa: que haga todo lo que se espera de él cada vez que se ejecuta. Un programa ha de escribirse de forma que, aunque usted no se halle presente cuando un amigo decida ejecutarlo por primera vez, funcione tal como se espera. Esto significa, por supuesto que no sólo está debidamente corregido sino que las instrucciones (que pueden estar contenidas en la subrutina de ASIGNACION DE VARIABLES) son claras y completas.

Las referencias del usuario no han de ofrecer ninguna duda para que el operador sepa con certeza si ha de poner un número, una serie de números, una palabra, una fecha, una combinación de letras y números, etc. El programa debe suponer que el operador es un completo idiota y que, por muy claras que sean las instrucciones y las referencias al usuario, siempre se intentará hacer las cosas incorrectamente. Un ejemplo clásico de esto es la entrada de fechas. Deben ponerse "trampas cazabobos" para rechazar una fecha cuando se pone de manera que no puede entender el ordenador (como el mes

antes que el día) o que sea claramente errónea (como, por ejemplo, el 32 de febrero). Hay que asegurar que cualquier cosa que haga el operador no interrumpa ni desvíe el programa. Ello puede ocurrir cuando se espera un dato numérico y el operador trata de poner una letra o una palabra; o pulsa ENTER sin haber aplicado nada previamente. Esto tiene la solución de permitir siempre una entrada de variable alfanumérica o cadena, volviendo a solicitar la entrada si se ha puesto una cadena vacía y tomando los enunciados VAL o CODE sobre la entrada para pasarla a forma numérica.

La documentación es una parte de la programación que con frecuencia se menosprecia. Resulta prácticamente imprescindible para un programa que se intenta publicar y muy aconsejable para los largos que se hacen para uno mismo. La documentación debe incluir, por lo menos, una lista de variables, una explicación de la estructura del programa (que será fácil de seguir si se aplica el enfoque modular que hemos aconsejado). También hay que incluir unas breves instrucciones, especialmente si el propio programa no las contiene. Asimismo resultan útiles una muestra de su ejecución indicando los datos de entrada esenciales, y la naturaleza y ordinograma del programa.

El programa debe ejecutarse con la máxima rapidez posible. Cada vez que aparece una subrutina y una remisión GO TO, el ordenador debe buscar, a través de todo el programa, línea por línea, hasta encontrar la que se solicita por lo que si se ponen las subrutinas de uso frecuente cerca del principio del programa, se acelerará su ejecución. Esta es la razón por la que las instrucciones se sitúan muy a menudo al final. No es conveniente que el ordenador tenga que recorrer la inicialización y las líneas con instrucciones cada vez que se le indique GO TO o GOSUB, en busca del destino que se le señala.

Defínanse primero las variables de uso frecuente para que ocupen los primeros lugares en el almacenamiento de variables. El ordenador busca sólo hasta que encuentra la que precisa por lo que ha de evitarse que extienda su búsqueda a más entradas que las necesarias.

Finalmente, llámese a un amigo y juntos siéntense frente al televisor, dígale solamente que pulse la tecla de ejecución (RUN) y observe. Si existe alguna vacilación o el programa no se ejecuta con suavidad, le queda trabajo aún por hacer. Esta es la mejor forma de comprobar un programa creado por uno mismo.

En resumen:

— Utilícense las instrucciones REM.
— Háganse los listados limpios y lógicos.

— Empléense las técnicas de programación estructurada, controlando el programa a través del bucle de llamada a las subrutinas.
— Examínese críticamente los GO TO incondicionales.
— Procúrese una presentación en pantalla atractiva y clara.
— Asegúrese que las referencias al usuario sean claras.
— Incorpórense "trampas cazabobos" en todas las entradas de datos para evitar aplicaciones incorrectas.
— Documéntense los programas aunque solamente se haga una lista de variables.
— Los programas deben ejecutarse lo más rápidamente posible.
— Pruébense los programas haciendo que los ejecute alguien que no esté familiarizado con ellos.

Programas, programas, programas

Finalmente presentamos algunos programas con los que se puede disfrutar.

```
   1 REM GALGOS
   2 REM © GOURLAY, HARTNELL 198
2
   5 RANDOMIZE
   7 GO SUB 200
  10 FOR X=1 TO 22
  20 PRINT INK 4;TAB 30;"▞"
  30 NEXT X
  35 PRINT AT 0,6;"APOSTAR SOBRE
 UN NUMERO ";W
  40 DIM A(9)
  50 FOR X=1 TO 9
  60 PRINT AT 2*X,A(X);"  "
  70 LET A(X)=A(X)+RND*2
  80 PRINT AT 2*X,A(X); INK X/2;
X
  85 BEEP .01,3*X
  90 IF A(X)>30 THEN GO TO 115
 100 NEXT X
 110 GO TO 50
 115 FOR G=1 TO 50 STEP 2
 120 PRINT AT 18,6; INK RND*7;X;
" ES EL GANADOR!"
```

```
 123 BEEP .02,G
 125 IF W=X THEN PRINT AT 20,3;
INK RND*7;"Y USTED TAMBIEN GANA!
!"
 130 FLASH RND
 140 NEXT G
 145 FLASH 0
 150 RUN
 200 BORDER 0
 205 PRINT AT 3,1; INK 2;"BIENVE
NIDO AL CANODROMO"
 210 PRINT AT 5,6; INK 4;"HAY NU
EVE PERROS."
 220 INPUT ( INK 2;"APUESTE POR
UN GANADOR         ");W
 230 IF W<1 OR W>9 THEN GO TO 22
0
 235 BORDER 2
 240 CLS : RETURN
```

```
  10 REM REBOTE DE ESTRELLAS
  15 REM © HARTNELL, 1982
  20 BORDER 0
  30 LET A=1: LET B=1: LET C=RND
*20: LET D=RND*20
  40 PRINT AT C,D;"▞";AT C,D; IN
K RND*7;"*"
  50 IF C+B>21 OR C+B<-1 THEN LE
T B=-B: BEEP 0.2,-RND*15
  60 IF D+A>31 OR D+A<0 OR RND>.
96 THEN LET A=-A: BEEP 0.1,RND*2
5
  70 LET C=C+B: LET D=D+A
  80 GO TO 40
```

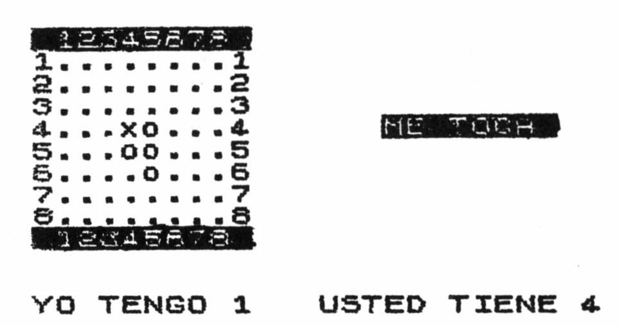
Color

Rete a su Spectrum con el juego que se propone en este programa "COLOR" escrito por Graham Charlton que pretende poner de relieve las posibilidades de sonido y color de su ordenador. Verá, cuando ejecute el programa, lo efectivas que son las características del Spectrum. El movimiento se efectúa entrando un número de la escala lateral seguido por otro del eje horizontal de la parte superior o inferior de la pantalla, como uno solo de dos cifras. Por ejemplo, si desea colocar una pieza donde se encuentra la "O" de abajo entraría el 64.

```
   1 REM ******* COLOR *********
   5 PRINT AT 0,12; INK 2;"c"; I
NK 1;"o"; INK 6;"l"; INK 3;"o";
INK 4;"u"; INK 5;"r"; INK 2;"t";
 INK 1;"h"; INK 6;"e"; INK 3;"l"
; INK 4;"l"; INK 2;"o"
  10 DIM a(10,10): FOR b=1 TO 10
: FOR c=1 TO 10
  20 BEEP .01,b*c/10
  40 IF b<>1 AND c<>1 AND b<>10
AND c<>10 THEN LET a(b,c)=CODE "
."
  50 NEXT c: NEXT b: LET p=0: LE
T r=0
  70 LET a(5,5)=CODE "x": LET a(
6,6)=CODE "x": LET a(6,5)=CODE "
o": LET a(5,6)=CODE "o"
 120 INPUT ( INK 2;"DESEA JUGAR
PRIMERO?  ");Q$
 125 CLS : GO SUB 3000
 127 PRINT AT 0,12; INK 2;"c"; I
NK 1;"o"; INK 6;"l"; INK 3;"o";
INK 4;"u"; INK 5;"r"; INK 2;"t";
 INK 1;"h"; INK 6;"e"; INK 3;"l"
; INK 4;"l"; INK 2;"o"
 130 IF CODE Q$<>CODE "n" AND CO
DE Q$<>CODE "n" THEN GO TO 2000
1000 PRINT INK 2;AT 10,16;"ME TO
CA"
1010 LET s=CODE "o": LET t=CODE
"x": LET h=0
1040 FOR a=2 TO 9: FOR b=2 TO 9
1060 IF a(a,b)<>CODE "." THEN GO
 TO 1320
1070 LET q=0: FOR c=-1 TO 1: FOR
 d=-1 TO 1: LET k=0: LET f=a: LE
T g=b
1130 IF a(f+c,g+d)<>s THEN GO TO
 1180
1140 LET k=k+1: LET f=f+c: LET g
=g+d: GO TO 1130
1180 IF a(f+c,g+d)<>t THEN GO TO
 1200
1190 LET q=q+k
1200 NEXT d
1210 NEXT c
1220 IF f=2 OR f=9 OR g=2 OR g=9
 THEN LET q=q*2
1230 IF f=3 OR f=8 OR g=3 OR g=8
 THEN LET q=q/2
1260 IF (f=2 OR f=9) AND (g=3 OR
 g=8) OR (f=3 OR f=8) AND (g=2 O
R g=9) THEN LET q=q/2
1280 IF q<h OR q=0 OR (RND>.3 AN
D q=h) THEN GO TO 1320
1290 LET h=q: LET m=a: LET n=b
1320 NEXT b
1330 NEXT a
1340 IF h=0 AND r=0 THEN GO TO 5
```

```
000
1350 IF h=0 THEN GO TO 1370
1360 GO SUB 4000: GO SUB 3000
2000 PRINT INK 1;AT 10,16;"SU TU
RNO"
2010 LET s=CODE "x": LET t=CODE
"o": INPUT r
2040 IF r=0 THEN GO TO 2090
2050 IF r<11 OR r>88 THEN GO TO
2030
2060 LET m=INT (r/10)+1: LET n=r
-10*INT (r/10)+1
2080 GO SUB 4000
2090 GO SUB 3000: GO TO 1000
3000 PRINT AT 5,0: BEEP .25,RND*
5
3010 LET c=0: LET h=0
3030 PRINT INK 4;" 12345678 "
3040 FOR b=2 TO 9: PRINT INK 4;b
-1;
3060 FOR d=2 TO 9
3070 IF a(b,d)=CODE "x" THEN PRI
NT INK 2;"x";
3075 IF a(b,d)=CODE "o" THEN PRI
NT INK 1;"o";
3077 IF a(b,d)=CODE "." THEN PRI
NT INK 5;".";
3080 IF a(b,d)=CODE "x" THEN LET
 c=c+1
3090 IF a(b,d)=CODE "o" THEN LET
 h=h+1
3100 NEXT d
3110 PRINT INK 4;b-1
3120 NEXT b
3130 PRINT INK 4;" 12345678 "
3150 PRINT '' INK 3;"YO TENGO ";
 INK 2;c; INK 3;"   USTED TIENE
"; INK 1;h;"   ",
3160 IF c+h=64 THEN GO TO 5000
3170 RETURN
4000 FOR c=-1 TO 1
4010 FOR d=-1 TO 1
4020 LET f=m: LET g=n
4040 IF a(f+c,g+d)<>s THEN GO TO
 4080
4050 LET f=f+c: LET g=g+d: GO TO
 4040
4080 IF a(f+c,g+d)<>t THEN GO TO
 4140
4090 LET a(f,g)=t: IF m=f AND n=
g THEN GO TO 4140
4110 LET f=f-c: LET g=g-d: GO TO
 4090
4140 NEXT d: NEXT c: RETURN
5000 IF c>h THEN PRINT "HE GANAD
O, ";c;" - ";h
5010 IF h>c THEN PRINT "USTED HA
 GANADO, ";h;" - ";c
5030 REM © CHARLTON 1982
```

colourthello



Programa "Vida"

Se presentan dos versiones del juego "Vida" (LIFE) de John Conway que simula el nacimiento, crecimiento y muerte de una colonia de células. Estas se desarrollan de acuerdo con las siguientes reglas:

— Cada célula en el reticulado tiene ocho vecinas.
— Cada célula, con dos o tres vecinas, sobrevive hasta la siguiente generación.
— Si sólo hay tres células vecinas, nace una nueva.
— Cualquier célula con cuatro o más vecinas muere por superpoblación.

```
  5 REM VIDA - © ANNE MARSHALL
 10 DIM A(145): DIM L(145): DIM
E(8)
 15 LET G=0
 20 FOR T=1 TO 8
 25 READ Z: LET E(T)=Z: NEXT T
 30 LET C=CODE "o": LET Z=128
 35 BORDER 1: PAPER 0: CLS
 40 FOR B=1 TO 12
 50 FOR D=1 TO 12
```

```
  60 LET A(B+10*D)=Z
  70 IF RND>.45 THEN LET A(B+10*
D)=C
  80 LET L(B+10*D)=A(B+10*D)
  90 NEXT D: NEXT B
 100 LET G=G+1
 120 FOR U=1 TO 12
 130 FOR B=1 TO 12
 140 LET F=U+10*B
 180 IF G=1 THEN GO TO 250
 190 LET H=0
 200 FOR T=1 TO 8
 210 IF A(F+E(T)+1)=C THEN LET H
=H+1
 220 NEXT T
 230 IF A(F)=C AND H<>3 AND H<>2
 THEN LET L(F)=Z
 235 IF A(F)=Z AND H=3 THEN LET
L(F)=C
 240 NEXT B: BORDER RND*7: NEXT
U
 245 BORDER 1
 250 FOR M=11 TO 144: LET A(M)=L
(M): BEEP .005,M/3: NEXT M
 255 PRINT AT 5,0;
 260 FOR U=1 TO 12: PRINT TAB 4;
 270 FOR B=1 TO 12: LET F=U+10*B
 280 PRINT INK 6;CHR$ A(F);" ";:
 NEXT B: PRINT : NEXT U
 285 PRINT AT 3,10; PAPER 2; INK
 6;"GENERACION ";G: BEEP .3,50
 290 GO TO 100
 300 DATA 11,10,0,1,-1,-9,-10,-1
1
```

```
  10 REM COLONIA DE CONWAY
  15 REM © HARTNELL, 1982
  20 GO SUB 90
  30 LET PRINT COLONIA=200
  40 LET GENERACION ACTUALIZADA=
320
  45 REM ************************
  50 GO SUB PRINT COLONIA
  60 GO SUB GENERACION ACTUALIZA
DA
  70 GO TO 50
  80 REM ************************
  90 REM INICIALIZA
 100 CLS
 110 LET CELULAS=0
 120 DIM A(11,11): DIM B(11,11)
 130 FOR X=2 TO 10: FOR Y=2 TO 1
0
 135 BORDER RND*7
 140 IF RND>.35 THEN LET A(X,Y)=
1: BEEP .02,X*Y/2: LET CELULAS=
CELULAS+1
```

```
 150 LET B(X,Y)=A(X,Y)
 160 NEXT Y: NEXT X
 170 LET A/O=0
 175 BORDER 7: CLS
 180 RETURN
 190 REM ***********************
 200 REM PRINT COLONIA
 210 LET A/O=A/O+1
 220 BEEP .02,RND*20
 250 PRINT AT 1,0; INK RND*6;"GE
NERACION>      ";A/O;AT 3,8;
 255 LET CELULAS=0
 260 FOR X=2 TO 10: FOR Y=2 TO 1
0
 270 LET A(X,Y)=B(X,Y)
 280 IF A(X,Y)=0 THEN PRINT " . "
;
 285 IF A(X,Y)=1 THEN PRINT INK
RND*6;"█ ";: LET CELULAS=CELULAS
+1
 290 NEXT Y: PRINT : PRINT : PRI
NT TAB 8;: NEXT X
 292 PRINT AT 21,0; INK RND*6;"C
ELULAS> ";CELULAS;"  "
 293 IF CELULAS<6 THEN RUN
 295 RETURN
 300 REM ***********************
 310 REM ACTUALIZACION
 340 FOR X=2 TO 10: FOR Y=2 TO 1
0
 345 BORDER RND*6
 350 LET C=0
 360 IF A(X-1,Y-1)=1 THEN LET C=
C+1
 370 IF A(X-1,Y)=1 THEN LET C=C+
1
 380 IF A(X-1,Y+1)=1 THEN LET C=
C+1
 390 IF A(X,Y-1)=1 THEN LET C=C+
1
 400 IF A(X,Y+1)=1 THEN LET C=C+
1
 410 IF A(X+1,Y-1)=1 THEN LET C=
C+1
 420 IF A(X+1,Y)=1 THEN LET C=C+
1
 430 IF A(X+1,Y+1)=1 THEN LET C=
C+1
 440 IF A(X,Y)=1 AND C<>2 AND C<
>3 THEN LET B(X,Y)=0
 450 IF A(X,Y)=0 AND C=3 THEN LE
T B(X,Y)=1
 460 NEXT Y: NEXT X
 470 RETURN
```

El juego de las cerillas

Se basa en el juego que apareció en la película "El Ultimo Año en Marienbad". Hay un cierto número de cerillas al comienzo del juego y entre el jugador y el ordenador, por turno, retiran una o más cada vez. El número máximo de las que se pueden tomar aparece en la parte superior de la pantalla. Quien retira la última de las cerillas pierde. El ordenador no es infalible.

```
   5 REM  * CERILLAS *
  10 REM TEXTO BLANCO EN AZUL
  15 PAPER 1: INK 7: BORDER 1: C
LS
  20 LET E=0: LET Z=16+INT (RND*
9)
  30 IF 2*(Z/2)=Z THEN LET Z=Z+1
  40 LET H=INT (RND*4)+2
  50 PRINT PAPER RND*5+2; INK 0;
AT 0,6;"EL MAXIMO A RETIRAR ES "
;H
  60 IF E>0 THEN PRINT AT 7,2;"U
STED TOMO ";E;TAB 20;"YO TOME ";
Q''
  70 FOR K=1 TO Z: BEEP .01,K
  80 PRINT INK RND*5+2;K;"█ ";
  90 IF RND>.85 THEN PRINT : PRI
NT
 100 NEXT K
 105 LET K=7: IF RND>.5 THEN LET
 K=4
 110 INPUT INK K;"CUANTAS TOMARA
? ";E
 120 IF E>H OR E<1 THEN GO TO 11
0
 130 CLS : LET Z=Z-E
 140 IF Z=0 THEN BORDER RND*7: P
RINT PAPER RND*5;AT 10,12;"YO GA
NO": BEEP .05,RND*30+30: GO TO 1
40
 150 LET Q=Z-1-INT ((Z-1)/(H+1))
*(H+1)+INT (RND*3)-1
 160 IF Q>Z OR Q<1 OR Q>H THEN G
O TO 150
 170 LET Z=Z-Q
 180 IF Z=0 THEN BORDER RND*7: P
RINT ; PAPER RND*6;AT 10,1;"YO T
OME ";Q;", ASI QUE GANA USTED!":
 BEEP .05,RND*40: GO TO 180
 190 GO TO 50
```

```
        EL MAXIMO A RETIRAR ES 4




     USTED TOMO 1        YO TOME 3
   1▌ 2▌ 3▌ 4▌ 5▌
   6▌ 7▌ 8▌ 9▌
   10▌ 11▌
```

Tragaperras

El programa siguiente le va a costar 1,50 dólares la tirada. De vez en cuando aparecerá la opción de PARO, pudiendo retener las cuatro piezas rotatorias si lo desea. En esta situación puede aplicar los números que quiere retener, pulsando ENTER después de cada entrada. Cuando ya tenga suficiente o no desee retener ninguno, pulse cinco y pulse ENTER, que le lleva otra vez a la siguiente jugada.

```
  20 POKE 23609,100
  30 GO SUB 9000
  40 POKE 23692,-1
  50 PAPER 0: CLS : BORDER 0: IN
K 7
  60 PRINT ''' PAPER 2;TAB 2;"ES
TA ES LA JUGADA ";JUGADA''TAB 2;
"USTED TIENE $";DINERO''TAB 2;"P
ULSE CUALQUIER TECLA PARA INICIA
R EL GIRO"
  70 IF INKEY$<>"" THEN GO TO 70
  80 IF INKEY$="" THEN GO TO 80
  85 POKE 23692,-1
  90 FOR G=1 TO 50: BORDER RND*7
: BEEP .01,50-G: NEXT G: BORDER
0
 100 FOR J=1 TO 4
 110 IF M(J)=J THEN GO TO 150
 120 LET A(J)=INT (RND*4)+1
 140 BEEP .1,50/J
 150 NEXT J
 165 LET JUGADA=JUGADA+1
```

```
 160 GO SUB 5000
 165 GO SUB 4000
 170 IF RND>.7 THEN GO SUB 6000
 176 FOR T=1 TO 40: PRINT AT 1,2
5; INK RND*7;"█ █";AT 1,25; INK
RND*7;" █ ": NEXT T
 177 FOR T=1 TO 25: PRINT : NEXT
T
 180 IF DINERO>0 THEN GO TO 60
 190 PRINT '''TAB 5;"SOBREVIVIO
";JUGADA;" JUGADAS"
 195 BORDER RND*7
 200 PRINT "PERO AHORA SE HA ARR
UINADO Y EL"
 205 BORDER RND*7
 210 PRINT "C A S I N O  E S T A
  C E R R A D O!"
 215 BORDER RND*7
 220 POKE 23692,-1
 230 PAUSE 10
 240 GO TO 190
4000 REM ** DINERO **
4005 PRINT ''': POKE 23692,-1
4010 LET DINERO=DINERO-1.5
4020 IF A(1)=A(2) AND A(2)=A(3)
AND A(3)=A(4) THEN PRINT INK 6;"
£££££  PREMIO!!!!  ££££££££££":
 BEEP 2,10: PRINT ''"GANA 10 DOL
ARES!!": LET DINERO=DINERO+10: G
O TO 4100
4030 IF (A(1)=A(2) AND (A(3)=A(2
) OR A(4)=A(2))) OR A(1)=A(2) AN
D A(2)=A(4) OR A(2)=A(3) AND A(3
)=A(4) THEN PRINT INK 6; PAPER 2
;"$$$$$$$ TRES IGUALES! $$$$$":
 BEEP 2,20: PRINT ''"GANA 5 DOLA
RES": LET DINERO=DINERO+5: GO TO
 4100
4040 IF A(3)=A(2) AND A(3)=A(4)
THEN PRINT INK 6; PAPER 2;"$$$$$
$$$ TRIO $$ TRIO $$ $$$$$": BEEP
 2,40: PRINT ''"GANA 7,5 DOLARES
": LET DINERO=DINERO+7.5: GO TO
4100
4050 IF A(1)+A(2)+A(3)+A(4)=10 T
HEN PRINT PAPER 2;">>>>>>>>>"; PA
PER 0;" ENHORABUENA!!": BEEP 2,-
30: PRINT ''"GANA 7,5 DOLARES!!"
: LET DINERO=DINERO+7.5
4100 FOR T=1 TO 20: BORDER RND*7
: BEEP .01,T: NEXT T: BORDER 0
4105 PRINT
4110 FOR T=1 TO 64: PRINT INK RN
D*7;"█";: NEXT T
4120 PRINT ''TAB 8;"USTED AHORA
TIENE $ ";DINERO''
4130 FOR T=1 TO 64: PRINT INK RN
D*7;"█";: NEXT T
```

```
4140 PRINT
4150 POKE 23692,-1: PRINT : PRINT
4160 DIM M(4)
4170 RETURN
5000 REM ** GIRO **
5010 FOR T=1 TO 50: BORDER RND*7: BEEP .01,50/T/2: NEXT T: BORDER 0
5020 PRINT ''''TAB 4;
5030 FOR J=1 TO 4
5040 IF A(J)=1 THEN PRINT INK 2;"█<>█ ";: BEEP .1,10
5050 IF A(J)=2 THEN PRINT INK 7;"█$$█ ";: BEEP .1,20
5060 IF A(J)=3 THEN PRINT INK 4;"█**█ ";: BEEP .1,30
5070 IF A(J)=4 THEN PRINT INK 5;"████ ";: BEEP .1,40
5080 PAUSE 70
5090 NEXT J
5100 RETURN
6000 REM ** RETENCION **
6010 DIM M(5)
6020 BEEP .5,1
6025 POKE 23692,-1
6030 PRINT '' INK 6;"PONGA LOS NUMEROS QUE"
6040 PRINT ' INK 6;"DESEE RETENER, ENTRE 5"
6050 PRINT ' INK 6;"CUANDO HAYA TERMINADO"
6060 INPUT Q
6070 IF Q<>5 THEN PRINT INK 2;Q
6080 LET M(Q)=Q
6090 IF Q<>5 THEN GO TO 6060
6100 RETURN
9000 REM ** ASIGNACION DE VARIABLES **
9010 DIM A(5): DIM M(5)
9020 LET DINERO=7.5
9030 LET JUGADA=1
9040 FOR T=1 TO 20
9050 BEEP .2,2*T
9060 NEXT T
9070 BORDER 7: PAPER 7: CLS
9080 BORDER 0: PAPER 0: CLS
9090 RETURN
```

El circuito final

Este juego fue adaptado de un programa del ZX80 (pista de carreras de 2K) que fue primeramente publicado por el Club Nacional de Usuarios del ZX de Gran Bretaña en la revista mensual INTERFACE. La versión original se debe a Alan Gunnell.

Resulta fácil de jugar y, como termina dando una puntuación después de cada carrera, constituye un reto para jugarlo repetidamente tratando de incrementar la puntuación. Hay tres pistas en las que se puede conducir con diversos grados de dificultad.

A lo largo de la carrera se le solicita para que ponga el valor de aceleración y la velocidad que desee. Pronto conocerá el efecto que tiene esto. Su puntuación se presenta constantemente (línea 220) y al final se da la total. Sus estímulos (incluyendo líneas como la del "EL CONDUCTOR DE DETRAS LE ESTA GRITANDO QUE ACELERE PORQUE LE ENTORPECE") se expresa en palabras en medio de la carrera. Observará que existe una gran tendencia a estrellarse y que su vehículo se las arregla para evitar un gran número de accidentes. De interés particular es la línea 290, que ocupa el lugar de cinco instrucciones IF/THEN del tipo IF H = 5 THEN LET B$ = = "recta con grasa", etc.

```
  5 REM CIRCUITO FINAL
 10 REM ADAPTADO DE UN PROGRAMA
 12 REM DEL ZX80 POR ALAN
    GUNNELL
 14 REM PUBLICADO PRIMERAMENTE
 16 REM       EN INTERFACE
 18 REM
 20 LET A=5: LET G=1: LET B=3
 22 BORDER 1: PAPER 7: INK 0
 25 INPUT "QUE PISTA DESEA (DE
LA 3 A LA 5)?";V
 27 IF V<3 OR V>5 THEN GO TO 25
 30 LET X=0
 40 LET L=100+V*V
 50 LET S=0
 60 IF X=10 THEN STOP
 80 LET X=X+1
 90 IF X=10 THEN PRINT INK RND*
6;TAB 8;"LA CARRERA HA TERMINADO
"'TAB 4;"LA PUNTUACION ES ";L;"
DE UN TOTAL DE ";100+V*V: POKE 2
3692,-1: BORDER RND*7: BEEP .02,
RND*30: GO TO 90
 110 GO SUB 180
```

```
 112 FOR T=1 TO 50: BEEP .02,T:
NEXT T
 115 GO SUB 270
 120 PRINT INK RND*6;B$
 125 GO SUB 145
 130 GO SUB 350
 135 PAUSE 50
 140 GO SUB 270
 142 GO TO 60
 145 FOR T=1 TO 50: BORDER RND*7
: NEXT T: BORDER 1
 149 LET S=ABS (S+(A*A)-(B*15)+(
2*G))
 150 PRINT '''' PAPER 2; INK 6;"
MARCHA ";G;"   VELOCIDAD ";S
 160 GO SUB H*1000
 180 BORDER 1: INPUT INK 7;"SELE
CCIONE UNA MARCHA (DE 1 A 10)";G
 190 IF G<1 OR G>10 THEN GO TO 1
80
 200 INPUT INK 7;"PONGA LA ACELE
RACION (DE 0 A 10)";A
 210 IF A<1 OR A>10 THEN GO TO 2
00
 220 PRINT INK RND*4+1;"LA PUNTU
ACION ACTUAL ES "; INK 2;L
 240 INPUT PAPER 2; INK 6;"ACCIO
NE EL FRENO (DE 0 A 10)";B
 250 IF B<0 OR B>10 THEN GO TO 2
40
 260 RETURN
 270 LET H=INT (RND*V)+1
 290 LET B$=("RECTA CON GRASA" A
ND H=5)+("CURVA CERRADA " AND H=
4)+("ESQUINA" AND H=3)+("CURVA"
AND H=2)+("RECTA" AND H=1)
 340 RETURN
 350 IF A=0 THEN LET A=1
 360 LET S=ABS (S+(A*A)-(B*15)+(
2*G))
 370 IF S<10 THEN LET S=10
 380 IF S<15 THEN PRINT ''''' IN
K 2;"EL CONDUCTOR DE ATRAS LE":
PRINT ' INK 1;"ESTA GRITANDO QUE
 ACELERE"
 400 RETURN
1000 IF S>90 THEN PRINT INK 2;"V
A MUY RAPIDO...REDUZCA!": LET L=
L-5
1010 RETURN
2000 IF S>40 THEN BEEP 3,50: FOR
 Q=1 TO 20: BORDER RND*6: NEXT Q
: PRINT INK 2;"CHOQUE"
2010 IF B>8 THEN PRINT INK 2;"CH
OQUE": BEEP .5,20: BORDER RND*6:
 PAUSE 20: LET L=L-9+INT (RND*10
)
2020 RETURN
```

```
3000 IF S>25 THEN FOR R=1 TO 10:
 PRINT INK RND*6;"CHOQUE!!!!!!!!!
!!": NEXT R: LET L=L-10
3010 RETURN
4000 IF S>35 THEN PRINT INK 2;"*
******CHOQUE*******************"
: LET L=L-10
4010 RETURN
5000 IF S>20 THEN PRINT INK 2;"C
CCCHHHOOOOOQQQUUUUEEEE!!!!!!!!"
: LET L=L-10
5005 FOR T=1 TO 50: PAPER RND*7:
 CLS : NEXT T: PAPER 7
5010 IF B>3 THEN PRINT "CHOQUE!!
!": LET L=L-10
5020 RETURN
```

El estallido

En este juego, basado en uno escrito por Eric Thompson, se controla la acción del pequeño trineo con las teclas "1" y "0". La bola se hace con la "o" minúscula.

```
   2 POKE 23609,100
   5 REM ESTALLIDO
   6 REM BASADO EN UN PROGRAMA
DEL ZX81
   7 REM POR ERIC THOMPSON
   9 GO SUB 500
  15 PAPER 6: CLS : BORDER 2
  20 FOR L=0 TO 7
  30 PRINT AT L,8; INK RND*3;"▚▚
▚▚▚▚▚▚▚▚▚▚▚▚▚▚"
  40 NEXT L
  55 LET S=0
  60 LET X=16
  70 LET B=INT (RND*10)+8
  80 LET Q=1-INT (RND*3)
  90 IF Q=0 THEN GO TO 80
 100 FOR P=8 TO -8 STEP -1
 110 IF ABS B>18 THEN LET B=18
 115 LET X=X+(INKEY$="0")-(INKEY
$="1")
 120 PRINT AT 9,X-2; INK 2;"  ▛▜
  ";AT P,B; INK 1;"o"
 140 LET B1=B
 150 IF B=8 OR B=18 THEN LET Q=-
Q: BEEP .005,30
```

```
 160 LET B=B+Q
 170 LET X=X+(INKEY$="0")-(INKEY
$="1")
 175 FOR G=1 TO Z: NEXT G
 180 PRINT AT P,B1;" "
 190 NEXT P
 195 LET X=X+(INKEY$="0")-(INKEY
$="1")
 200 IF ABS (B-X)>2 THEN GO TO 2
30
 205 LET S=S+1
 210 PRINT AT 16,8; INK 1;"SU PU
NTUACION ES "; INK 2;S: BEEP .25
,1
 220 GO TO 80
 230 FOR G=1 TO 400: NEXT G
 240 LET S=0
 250 GO TO 15
 500 INPUT "GRADO DE DIFICULTAD
(DEL 1 AL 20)? ";Z
 510 IF Z<1 OR Z>20 THEN GO TO 5
00
 520 LET Z=2*Z
 530 FOR S=1 TO Z
 540 BEEP .01,S
 550 NEXT S
 560 RETURN
```

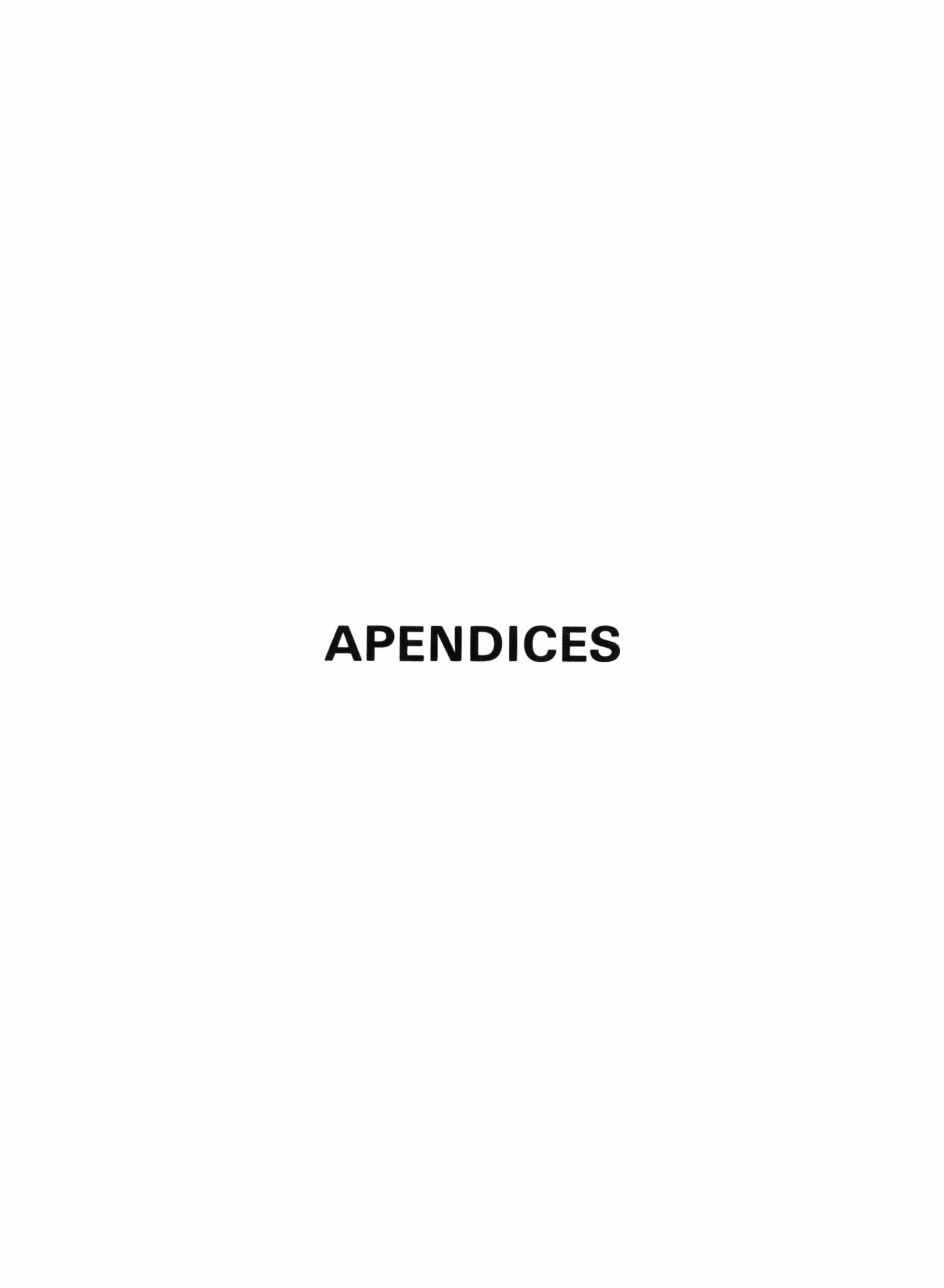
"GALXIAN"

Este programa fue adaptado por Tim Hartnell de uno del ZX81 escrito por James y Paul Holmes y se publicó primeramente en la revista DATABUS.

```
  1 REM        GALXIAN  -
     HOLMES, WALSH, HARTNELL
  4 GO SUB 1000
  5 BORDER 0
  6 PAPER 0: CLS
  7 INK 6
 10 LET X=10
 20 LET S=0
 30 LET P=1
 40 LET S=S+P
 50 CLS
 60 PRINT AT 0,0;S
 70 LET P=0
```

```
  80 LET N=10
  90 LET Y=13
 100 LET C=0
 110 PRINT AT 15,X;"   "
 120 IF INKEY$="5" THEN LET X=X-
1
 130 IF INKEY$="8" THEN LET X=X+
1
 140 PRINT AT 15,X; INK 2;A$
 145 BEEP .004,10
 150 PRINT AT P,N;" "
 160 IF RND>.65 THEN LET N=N+INT
 (RND*3-1)
 170 IF RND>.8 THEN LET P=P+1
 190 PRINT AT P,N; INK 4;"[illegible]"
 200 IF P>14 THEN GO TO 5
 210 IF C=0 AND INKEY$="1" THEN
LET C=X+1
 220 IF C=0 THEN GO TO 110
 230 PRINT AT Y,C;" "
 240 LET Y=Y-1
 250 IF Y<4 THEN GO TO 90
 260 PRINT AT Y,C;"[illegible]": BEEP .009
,50
 270 IF C=N AND Y=P THEN FOR W=1
 TO 3: PRINT AT Y,C; INK RND*8;"
*": BEEP .15,15: BORDER RND*7: B
EEP .15,20: BORDER RND*7: BEEP .
15,30: NEXT W: GO TO 40
 280 GO TO 110
1000 LET A$="[illegible]"
1005 FOR J=0 TO 7
1010 READ N
1020 POKE USR "A"+J,N
1030 NEXT J
1040 FOR J=0 TO 7
1050 READ N
1060 POKE USR "B"+J,N
1070 NEXT J
1080 FOR J=0 TO 7
1090 READ N
1100 POKE USR "C"+J,N
1110 NEXT J
1120 FOR J=0 TO 7
1130 READ N
1140 POKE USR "M"+J,N
1150 NEXT J
1160 FOR J=0 TO 7
1170 READ N
1180 POKE USR "E"+J,N
1190 NEXT J
2000 DATA BIN 11111110,BIN 11111
100,BIN 11111000,BIN 11110001,BI
N 01100001,BIN 10010001,BIN 1011
1000,BIN 10111100
2010 DATA BIN 01111110,BIN 10111
101,BIN 11000011,BIN 01000010,BI
N 00000000,BIN 01000010,BIN 0011
1100,BIN 00111100
```

```
2020 DATA BIN 01111111,BIN 00111
111,BIN 00011111,BIN 10001111,BI
N 10000110,BIN 10000001,BIN 0001
1101,BIN 00111101
2030 DATA BIN 01111110,BIN 10011
001,BIN 11000011,BIN 01100110,BI
N 01100110,BIN 10000001,BIN 1100
0011,BIN 11100111
2040 DATA BIN 11110111,BIN 11100
011,BIN 11010101,BIN 10110110,BI
N 11110111,BIN 11110111,BIN 1110
0011,BIN 11001001
5000 RETURN
```



APENDICES

Sistema de memoria en microdiscos ("MICRODRIVE")

Este sistema miniatura de memoria en discos flexibles ("microfloppy") alberga hasta 100K de programa o datos, pudiendo conectarse hasta ocho de ellos simultáneamente en el Spectrum. El ritmo de transferencia de información desde el sistema hasta el ordenador es de 16K por segundo y el tiempo total de exploración para todo el disco hasta encontrar un determinado punto es de siete segundos, aunque muchos tiempos de acceso serán inferiores a éste. Algunas instrucciones que no se han tratado en este libro tienen su aplicación enteramente en el "Microdrive" y en el sistema de interfaz RS232.

El interfaz RS232

Este sistema permite que el Spectrum se conecte a cualquier periférico (como impresora, terminales, otros ordenadores, etc.) que sea compatible con él. El RS232 es un standard adoptado por la industria por lo que existe una amplia gama de aplicaciones para el Spectrum asociado a este interfaz. El sistema del interfaz se encuentra en el monitor del Spectrum.

Otras instrucciones

Los enunciados OPEN#, CLOSE#, MOVE, ERASE, CAT y FORMAT se utilizan con el interfaz y con el "Microdrive". El "Microdrive" admite no sólo SAVE, VERIFY, LOAD Y MERGE sino también PRINT, LIST, INPUT e INKEY$.

IN y OUT son instrucciones para accionar las entradas y salidas del ordenador y sirven para controlar u obtener información del teclado o de la impresora.

Conversión binaria a decimal

Puede preferirse utilizar la numeración decimal a la binaria en las indicaciones de datos para los gráficos definidos por el usuario. En este caso será útil la lista que sigue. Por si le interesa, le indicamos que el programa utilizado para imprimir esta lista se da al final.

00000000	0
00000001	1
00000010	2
00000011	3
00000100	4
00000101	5
00000110	6
00000111	7
00001000	8
00001001	9
00001010	10
00001011	11
00001100	12
00001101	13
00001110	14
00001111	15
00010000	16
00010001	17
00010010	18
00010011	19
00010100	20
00010101	21
00010110	22
00010111	23
00011000	24
00011001	25
00011010	26
00011011	27
00011100	28
00011101	29
00011110	30
00011111	31
00100000	32
00100001	33
00100010	34
00100011	35
00100100	36
00100101	37
00100110	38
00100111	39
00101000	40
00101001	41
00101010	42
00101011	43
00101100	44
00101101	45
00101110	46
00101111	47
00110000	48
00110001	49
00110010	50
00110011	51
00110100	52
00110101	53
00110110	54
00110111	55
00111000	56
00111001	57
00111010	58
00111011	59
00111100	60
00111101	61
00111110	62
00111111	63
01000000	64
01000001	65
01000010	66
01000011	67
01000100	68
01000101	69
01000110	70
01000111	71
01001000	72
01001001	73
01001010	74
01001011	75
01001100	76
01001101	77
01001110	78
01001111	79
01010000	80
01010001	81
01010010	82
01010011	83
01010100	84
01010101	85
01010110	86
01010111	87
01011000	88
01011001	89
01011010	90
01011011	91

01011100	92
01011101	93
01011110	94
01011111	95
01100000	96
01100001	97
01100010	98
01100011	99
01100100	100
01100101	101
01100110	102
01100111	103
01101000	104
01101001	105
01101010	106
01101011	107
01101100	108
01101101	109
01101110	110
01101111	111
01110000	112
01110001	113
01110010	114
01110011	115
01110100	116
01110101	117
01110110	118
01110111	119
01111000	120
01111001	121
01111010	122
01111011	123
01111100	124
01111101	125
01111110	126
01111111	127
10000000	128
10000001	129
10000010	130
10000011	131
10000100	132
10000101	133
10000110	134
10000111	135
10001000	136
10001001	137
10001010	138
10001011	139
10001100	140
10001101	141
10001110	142
10001111	143
10010000	144
10010001	145
10010010	146
10010011	147
10010100	148
10010101	149
10010110	150
10010111	151
10011000	152
10011001	153
10011010	154
10011011	155
10011100	156
10011101	157
10011110	158
10011111	159
10100000	160
10100001	161
10100010	162
10100011	163
10100100	164
10100101	165
10100110	166
10100111	167
10101000	168
10101001	169
10101010	170
10101011	171
10101100	172
10101101	173
10101110	174
10101111	175
10110000	176
10110001	177
10110010	178
10110011	179
10110100	180
10110101	181
10110110	182
10110111	183
10111000	184
10111001	185
10111010	186
10111011	187
10111100	188
10111101	189
10111110	190
10111111	191
11000000	192
11000001	193
11000010	194
11000011	195
11000100	196
11000101	197
11000110	198
11000111	199
11001000	200
11001001	201
11001010	202
11001011	203
11001100	204
11001101	205
11001110	206
11001111	207
11010000	208
11010001	209

11010010	210
11010011	211
11010100	212
11010101	213
11010110	214
11010111	215
11011000	216
11011001	217
11011010	218
11011011	219
11011100	220
11011101	221
11011110	222
11011111	223
11100000	224
11100001	225
11100010	226
11100011	227
11100100	228
11100101	229
11100110	230
11100111	231
11101000	232
11101001	233
11101010	234
11101011	235
11101100	236
11101101	237
11101110	238
11101111	239
11110000	240
11110001	241
11110010	242
11110011	243
11110100	244
11110101	245
11110110	246
11110111	247
11111000	248
11111001	249
11111010	250
11111011	251
11111100	252
11111101	253
11111110	254
11111111	255

```
 10 FOR B=0 TO 255
 20 LET J=B
 30 LET A$=""
 40 FOR N=0 TO 7
 50 LET T=J-INT (J/2)*2
 60 IF T=0 THEN LET A$="0"+A$
 70 IF T<>0 THEN LET A$="1"+A$
 80 LET J=INT (J/2)
 90 NEXT N
100 PRINT A$,B
110 NEXT B
```

Mensajes codificados

Estos mensajes aparecen en el fondo de la pantalla siempre que se detiene el ordenador durante la ejecución de un programa en BASIC. Explican el porqué de la detención tanto si es por causa natural o debido a un error.

La instrucción de Continuar (CONTINUE) remite el programa a la línea e instrucción especificada en el último mensaje, aunque hay excepciones con relación a los códigos 0, 9 y D.

Significación de los códigos

0	OK (Todo correcto). Terminación correcta o salto a un número de línea superior a cualquiera de los existentes.
1	Existe una instrucción NEXT sin la FOR con ella relacionada.
2	Variable no encontrada. Un subíndice.
3	Indice equivocado. El superior a las dimensiones del conjunto o hay un número equivocado de subíndices.
4	Falta memoria.
5	Fuera de la pantalla. INPUT ha tratado de generar más de 23 líneas en la mitad inferior de la pantalla. Ocurre lo mismo con PRINT AT 22, ...
6	Número demasiado grande. Los cálculos han conducido a una cantidad superior a 10^{38}.
7	Hay una instrucción RETURN sin existir el GOSUB con el que ha de estar relacionada.
8	Fin de archivo. De aplicación en las operaciones con Microdrive, interfaz, etc.
9	PARO.
A	Argumento no válido.
B	Entero fuera de rango.
C	Sin sentido en BASIC
D	PARO. CONT repite la instrucción donde se paró el programa. El comportamiento de CONTINUE después de este mensaje es normal en cuanto repite la instrucción donde se produjo el "BREAK".

E Faltan datos.
F Nombre de archivo no válido.
G No hay espacio para la línea.
H Paro en la Entrada.
I FOR sin el correspondiente NEXT.
J Sistema ENTRADA/SALIDA no válido, en operaciones con periféricos: Microdrive, etc.
K Color no válido.
L Corte del programa.
Se ha pulsado el comando BREAK. Esto se detecta entre dos instrucciones.
M Dirección más alta de la memoria RAM errónea. El número especificado por RAMTOP es demasiado grande o demasiado pequeño.
N Enunciado perdido.
Salto a una instrucción que ya no existe.
O Flujo de datos o información no válidos en operaciones con periféricos: Microdrive, etc.
P Función sin el prefijo DEF (Definición).
Q Error en el parámetro.
Número erróneo de argumentos, o uno de ellos de tipo equivocado (cadena alfanumérica en lugar de variable numérica o viceversa).
R Error de carga de la cinta magnética.

OBRAS DE EDICIONES TECNICAS REDE

ACUSTICA Y BAJA FRECUENCIA

ALTA FIDELIDAD A BAJO COSTE (Libro n.° 87)
212 págs., 117 figs. Redacción REDE.

AUDIO REPARACION (Libro n.° 126)
170 págs., 187 figs. Autor: F. Mor.

COMUNICACION INSTANTANEA (Libro n.° 109)
114 págs., 54 figs. Redacción REDE.

CONSTRUCCION PRACTICA Y SIMPLIFICADA DE CAJAS ACUSTICAS PARA HI-FI (Libro n.° 138)
122 págs., 79 figs. Redacción REDE.

INICIACION AL DISEÑO DE CIRCUITOS DE AUDIO (Libro n.° 148)
108 págs., 42 figuras. Redacción REDE.

MAGNETOFONOS Y CASSETTES (Libro n.° 151)
168 págs., 75 figuras. Redacción REDE.

MUSICA ELECTRONICA (Libro n.° 83)
150 págs., 70 figs. Redacción REDE.

CIRCUITOS COMPROBADOS

Un tipo de libro sin precedentes. Cada montaje, antes de ser descrito, ha sido realizado y mantenido en óptimo funcionamiento en el Laboratorio de Experimentación de REDE. Tamaño de cada volumen: 210 × 270 mm.

AUDIO-1 (Libro n.° 111)
84 págs., 100 figs. Redacción REDE.

MONTAJES PRACTICOS (Libro n.º 115)
84 págs., 121 figs. Redacción REDE.

PRACTICA DIGITAL (Libro n.º 118)
80 págs., 134 figs. Redacción REDE.

CIRCUITOS INTEGRADOS (Libro n.º 128)
84 págs. 120 figs. Redacción REDE.

AUDIO-2 (Libro n.º 131)
88 págs., 100 figs. Redacción REDE.

JUEGOS ELECTRONICOS-I (Libro n.º 132)
80 págs. 100 figs. Redacción REDE.

ANTI-ROBO (Libro n.º 135)
76 págs., 114 figs. Redacción REDE.

JUEGOS ELECTRÓNICOS-II (Libro n.º 141)
80 págs., 96 figs. Redacción REDE.

AUDIO-3 (Libro n.º 143)
82 págs., 116 figs. Redacción REDE.

TELEMANDO (Libro n.º 146)
80 págs. 112 figs. Redacción REDE.

COMPROBADORES (Libro n.º 152)
84 págs. 100 figuras. Redacción REDE.

AUDIO-4 (Libro n.º 154)
80 págs., 100 figs. Redacción REDE.

ELECTRONICA EN EL AUTOMOVIL (Libro n.º 158)
84 pág., 96 figs. Redacción REDE.

LUCES SICODELICAS Y JUEGOS LUMINOSOS
(Libro n.º 160)
82 págs., 100 figs. Redacción REDE.

COMPONENTES ELECTRONICOS

EL TIRISTOR: APLICACIONES, CARACTERISTICAS Y FUNCIONAMIENTO
(Libro n.° 73)
104 págs., 50 figs. Autor: R. Swoboda.

LA FIABILIDAD DE LOS COMPONENTES ELECTRÓNICOS (Libro n.° 70)
162 págs., 28 figs., 31 tablas. Autor: C. E. Jowet.

ELECTRICIDAD Y MEDICION

INSTALACIONES ELECTRICAS DE BAJA TENSION
(Libro n.° 64)
136 págs. Autores: A. Bandini y M. Bertolini.

LUMINOTECNIA (Libro n.° 58)
177 págs., 87 figs., 46 tablas. Autor: G. Clerici.

MANTENIMIENTO DE EQUIPOS ELECTRICOS
(Libro n.° 14)
111 págs., 82 figs., 20 tablas. Autor: P. L. Cerato.

MEDIDAS ELECTRICAS (I/I): METODOS E INSTRUMENTOS (Libro n.° 57)
328 págs., 194 figs. 50 tablas. Autor: A. Bandini.

MEDICION ELECTRICA (I/II): METODOS E INSTRUMENTOS (Libro n.° 62)
288 págs., 153 figs. 42 tablas. Autor: A. Bandini.

MEDICION ELECTRICA (II): ENSAYOS DE MAQUINAS (Libro n.° 59)
400 págs., 120 figs. Autores: A. Bandini y M. Bertolini.

METODOS DE MEDIDA EN CIRCUITOS DE CORRIENTE CONTINUA (Libro n.° 54)
139 págs., 90 figs. Autor: A. Bossi.

SEÑALIZACIONES ELECTRICAS
(Libro n.° 32) 226 págs., 132 figs. Autor: G. Clerici.

ESQUEMARIOS

Cada volumen constituye un elemento insustituible tanto para el reparador de TV b/n y color como para los especializados en magnetófonos y cassettes, incluyendo esquemas y notas de servicio.

TV Blanco y negro

ESQUEMARIO TV/I	(libro n.° 21)
ESQUEMARIO TV/II	(libro n.° 22)
ESQUEMARIO TV/III	(libro n.° 55)
ESQUEMARIO TV/IV	(libro n.° 67)
ESQUEMARIO TV/V	(libro n.° 76)
ESQUEMARIO TV/VI	(libro n.° 81)
ESQUEMARIO TV/VII	(libro n.° 88)
ESQUEMARIO TV/VIII	(libro n.° 94)
ESQUEMARIO TV/IX	(libro n.° 100)
ESQUEMARIO TV/X	(libro n.° 105)
ESQUEMARIO TV/XI	(libro n.° 113)
ESQUEMARIO TV/XII	(libro n.° 127)
ESQUEMARIO TV/XIII	(libro n.° 140)
ESQUEMARIO TV/XIV	(libro n.° 155)

TV Color

ESQUEMARIO TVC/I	(libro n.° 112)
ESQUEMARIO TVC/II	(libro n.° 114)
ESQUEMARIO TVC/III	(libro n.° 116)
ESQUEMARIO TVC/IV	(libro n.° 117)
ESQUEMARIO TVC/V	(libro n.° 119)
ESQUEMARIO TVC/VI	(libro n.° 129)
ESQUEMARIO TVC/VII	(libro n.° 134)
ESQUEMARIO TVC/VIII	(libro n.° 136)
ESQUEMARIO TVC/IX	(libro n.° 137)
ESQUEMARIO TVC/X	(libro n.° 142)
ESQUEMARIO TVC/XI	(libro n.° 145)
ESQUEMARIO TVC/XII	(libro n.° 150)
ESQUEMARIO TVC/XIII	(libro n.° 157)
ESQUEMARIO TVC/XIV	(libro n.° 161)
ESQUEMARIO TVC/XV	(libro n.° 162)
ESQUEMARIO TVC/XVI	(libro n.° 165)

Magnetófonos y Cassettes

ESQUEMARIO I	(libro n.° 85)	**ESQUEMARIO V**	(libro n.° 133)
ESQUEMARIO II	(libro n.° 103)	**ESQUEMARIO VI**	(libro n.° 139)
ESQUEMARIO III	(libro n.° 123)	**ESQUEMARIO VII**	(libro n.° 144)
ESQUEMARIO IV	(libro n.° 130)	**ESQUEMARIO VIII**	(libro n.° 159)

LIBROS-HERRAMIENTA

Obras destinadas al reparador de TV, radio, auto-radio, electrodomésticos, magnetófonos, etc., al montador y al experimentador.

ALARMA ELECTRONICA (libro n.° 102)
150 págs. 91 figs. Redacción REDE.

AUTOMATISMOS DE FACIL CONSTRUCCION
(Libro n.° 107)
128 págs. 89 figs. Redacción REDE.

BIOELECTRONICA (Libro n.° 149)
132 págs., 72 figs. Redacción REDE.

COMODIDADES ELECTRONICAS DE FACIL MONTAJE
(Libro n.° 99)
124 págs. 75 figs. Redacción REDE.

CONTRAESPIONAJE ELECTRONICO (Libro n.° 93)
115 págs., 50 figs. Redacción REDE.

ELECTRONICA EN LA FOTOGRAFIA (libro n.° 121)
126 págs., 52 figs. Redacción REDE.

ELECTRONICA AL SERVICIO DEL AUTOMOVILISTA, LA (Libro n.° 122)
146 págs. 68 figs. Redacción REDE.

ESPIONAJE ELECTRONICO (Libro n.° 84)
128 págs. 62 figs. Redacción REDE.

IMPROVISACIONES QUE DAN DINERO Y AHORRAN TIEMPO
Volumen I (libro n.° 48): 140 págs., 91 figs.
Volumen II (libro n.° 63): 126 págs., 77 figs.
Volumen III (libro n.° 80): 130 págs., 82 figs.
Volumen IV (libro n.° 98): 146 págs., 90 figs.
Volumen V (libro n.° 104): 138 págs., 91 figs.

JUGUETES ELECTRÓNICOS - I (Libro n.° 96)
122 págs., 61 fig. Redacción REDE.

JUGUETES ELECTRONICOS - II (Libro n.° 106)
128 págs. 91 figs. Redacción REDE.

PRACTICA ELECTRONICA SIMPLIFICADA Y EXPERIMENTAL: CON 1 TRANSISTOR, MULTIPLES MONTAJES COMPROBADOS
(Libro n.° 120)
118 págs. 62 figs. Redacción REDE.

PRACTICA ELECTRONICA SIMPLIFICADA Y EXPERIMENTAL: CON 2 TRANSISTORES, MULTIPLES MONTAJES COMPROBADOS
(Libro n.° 124)
140 págs. 67 figs. Redacción REDE.

PRACTICA ELECTRONICA SIMPLIFICADA Y EXPERIMENTAL: CON 3 TRANSISTORES, MÚLTIPLES MONTAJES COMPROBADOS
(Libro n.° 125)
132 págs. 64 figs. Redacción REDE.

RECUPERACIÓN DE COMPONENTES ELECTRONICOS
(Libro n.° 156)
166 págs., 91 figs. Redacción REDE.

REPARACION DE ELECTRODOMESTICOS
(Libro n.° 40)
265 págs., 167 figs. Autor: E. Tricomi.

SOLDADURA ELECTRICA EN LOS MONTAJES ELECTRONICOS (Libro n.° 147)
104 págs., 69 figs. Autor: F. Mor.

SEGURIDAD ELECTRONICA (Libro n.° 108)
120 págs. 60 figs. Redacción rede.

UHF, TECNICA/ADAPTACION/REPARACION
(Libro n.° 33)
335 págs. 302 figs. Autor: F. Möhring.

TELEVISION

PRACTICA DE LA CONSTRUCCION E INSTALACION DE ANTENAS DE FM Y DE TV (Libro n.° 92)
272 págs. 222 figs. Redacción REDE.

REPARACION TV-I (Libro n.° 77)
300 págs. 472 figs. Autor: F. Mor.

REPARACION TV-II (Libro n.° 110)
304 págs. 470 figs. Autor: F. Mor.

REPARACION TVC (Libro n.° 153)
140 págs. Ilustraciones a todo color. Redacción REDE.

TELEVISION COLOR BASICA (Libro n.° 166)
170 págs., 80 figs. Ilustraciones a todo color.
Autor: B. Miguel.

VARIOS

CONTROL NUMERICO DE LAS MAQUINAS HERRAMIENTAS (Libro n.° 86)
90 págs., 64 figs. Autor: M. Flego.

DIBUJO INDUSTRIAL (Libro n.° 35)
260 págs., 158 figs., 56 tablas. C. Clerici.

DIBUJO TECNICO (Libro n.° 15)
154 págs., 82 figs., 20 tablas. C. Clerici.

Solicite el catálogo a
EDICIONES TECNICAS REDE
Apartado 35400 - Barcelona

FE DE ERRATAS

LA MEJOR PROGRAMACIÓN DEL ZX SPECTRUM POR LA PRÁCTICA

ERRATAS Y ACLARACIONES

Pág. 17. — En el programa «Tabulador lanzamiento de cohetes», la línea 20, por defecto de impresión, puede parecer que indique:

```
20 DIM A$(5,6)
```

en realidad, su lectura correcta es:

```
20 DIM A$(6,6)
```

Pág. 19. — En el párrafo central de la página, situado encima del título «Conservación de programas», en la línea quinta, se dice:

utiliza un número aleatorio de veces determinado por 9

ha de decir:

utiliza un número aleatorio de veces determinado por q

Pág. 104. — En el segundo programa — REM CONJUNTOS MULTIDIMENSIONALES — la línea 120 se presenta así:

```
 120 PRINT TAB 13;B,TAB 15;A(B,1
);" ";A(B,2);" ";A(B,3);" ";A(B,4)
```

ha de presentarse así:

```
 120 PRINT TAB 13;B;TAB 15;A(B,1
);" ";A(B,2);" ";A(B,3);" ";A(B,4)
```

Pág. 105. — En el primer párrafo, la penúltima línea empieza así:

en el número 2

Ha de empezar así:

en el número 5

La última línea del mismo párrafo acaba así:

es igual a 2

Ha de acabar así:

es igual a 5

Pág. 117. — En la última línea se dice:

Pruebe a aplicar VAL a una expresión como "ATN 1 x 4"

Ha de decir:

Pruebe a aplicar VAL a una expresión como "ATN 1 * 4"

Pág. 147. — En el segundo párrafo se dice:

Una ventaja es que la M

ha de decir:

Una desventaja es que la M

Pág. 172. — En el listado incluido en esta página, la línea 265 se presenta así:

```
 265 PRINT "ENTRAR LOS DATOS DES
PUES DE      PULSAR NEWLINE"
```

Ha de presentarse así:

```
 265 PRINT "ENTRAR LOS DATOS PUL
SANDO ENTER"
```

ediciones técnicas
REDE
BARCELONA
(ESPAÑA)